NASA CR-
147707

REPORT NO. 26960-6001-TU-00

(NASA-CR-147707) ZERO-G FLIGHT TEST OF A GAUGING SYSTEM. VOLUME 2: SYSTEM SOFTWARE (TRW Systems Group) 83 p HC $5.00 CSCL 14B

N76-23350

Unclas
G3/19 28152

ZERO-G FLIGHT TEST OF A GAUGING SYSTEM

VOLUME II: SYSTEM SOFTWARE

CONTRACT NO. NAS 9-14349
JANUARY 1976

PREPARED FOR:
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
LYNDON B. JOHNSON SPACE CENTER
HOUSTON, TEXAS 77058

RECEIVED
NASA STI FACILITY
INPUT BRANCH

TRW
DEFENSE AND SPACE SYSTEMS GROUP

REPORT NO. 26960-6001-TU-00

ZERO-G FLIGHT TEST OF A GAUGING SYSTEM

VOLUME II: SYSTEM SOFTWARE

CONTRACT NO. NAS 9-14349
JANUARY 1976

PREPARED FOR:
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
LYNDON B. JOHNSON SPACE CENTER
HOUSTON, TEXAS 77058

TRW
DEFENSE AND SPACE SYSTEMS GROUP

FOREWORD

This technical report concludes Contract No. NAS 9-14349, "Zero-G Flight Test of a Gauging System," and covers the experimental flight testing of a nucleonic quantity gauging system conducted on-board a KC-135 Zero-G Aircraft. The report was prepared by Frank E. Bupp of TRW Defense and Space Systems Group, One Space Park, Redondo Beach, California. The author is grateful to numerous personnel for assistance in completion of this program, particularly to Mr. M. McFarlin of TRW, who was responsible for the development of the flight system software and data reduction and analysis, and Mr. R. Cardon of TRW for electronic support. The author also expresses his grateful appreciation to Mr. J. Alexander of the NASA/JSC and to Mr. D. Griggs, Chief, Zero-G Project Office, and the members of the flight crew for their contributions and dedicated participation in the conduct of the flight tests.

This report contains no classified information extracted from other classified reports; this report was submitted in January 1976.

CONTENTS

CONTENTS (Continued)

ILLUSTRATIONS

ILLUSTRATIONS (Continued)

TABLES

SECTION I

INTRODUCTION

Since November of 1970, a nucleonic gauging system for gauging propellant in tankage under zero-gravity conditions has been under development at TRW Systems. This gauging system is based upon the principle of "X-raying" the interior of the tank wherein the absorption of the diagnostic radiation by any intervening propellant is utilized to determine the propellant mass. The technological feasibility of this diagnostic technique was first demonstrated successfully in June of 1970 in the form of a breadboard model installed on battleship tankage under one-g conditions. In the interest of minimizing cost, this early feasibility model was interfaced with a large, general purpose computer for requisite data processing. The results obtained with this early breadboard model were, to say the least, promising, particularly in the high levels of accuracy that were achieved. Consequently, a complete flight-type system including an integral microcomputer was next designed, fabricated, and subsequently tested in November of 1973. This flight-type system, described in detail in Reference 1, was highly successful and achieved total system accuracies of 0.35 percent and better; of fundamental importance to this high level of accuracy, however, was the system software (i.e., requisite to the mass computation). This report, therefore, documents the nucleonic gauging system software for the computation of propellant mass; this software a Zero-G Gauging System," and subsequently modified to meet the requirements of zero-g flight testing under Contract No. NAS 9-14349.

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

SECTION II

GENERAL DESCRIPTION OF THE NUCLEONIC GAUGE CONCEPT

In its most general form, a nucleonic gauge is comprised of an array of gamma ray emitting radioisotope sources positioned on one side of a (propellant) tank, and an array of radiation detectors positioned on the other side of the tank opposite the sources; the concept is illustrated in Figure 2-1. Although all the elements and subsystems of the gauge are external to the tank, gamma rays that are emitted toward the detectors will penetrate the tank walls and be partially absorbed by the intervening fluid (fuel) within the tank. The amount of radiation which emerges on the detector side of the tank is then proportional to the mass of the intervening propellant.

In practice, there are two basically different methods of mechanizing a nucleonic gauge: one mechanization is based on the natural – i.e., exponential – gamma ray absorption characteristics of the propellant while the other is based on obtaining a linear gamma ray absorption relationship. The latter technique is accomplished by selectively employing a window discriminator in the detection circuitry to accept only those photons which fall within a narrow energy range, thus making the gamma ray absorption a linear function of the absorber thickness. Although the linear system is the easier of the two systems to construct (at least from a hardware point of view), it is severely limited to tank dimensions the order of 8 feet or less; this limitation precluded the linear system from further consideration for the envisioned application and hence it was not subjected to rigorous study during this program. This report discusses only those systems based on exponential absorption.

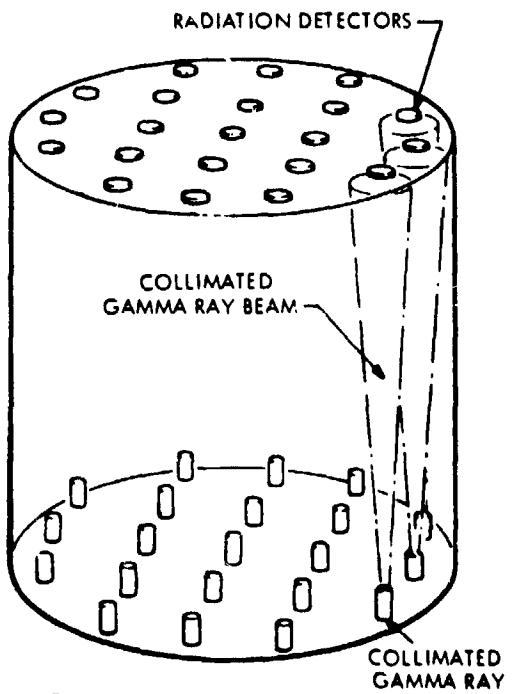


Figure 2-1. Typical Source-Detector Configuration

For a propellant gauge employing the exponential principle of gamma ray absorption to operate properly under low Bond number regimes and/or zero-g conditions, it must first be insensitive to the fuel orientation within the tank. This is accomplished by judiciously distributing the detectors and radioisotope sources opposite each other and assuring that the detectors and sources are configured into a suitable array which covers a substantial portion of the entire projected area of the tank. The source-detector array is absolutely necessary unless the fuel is constrained by some means (such as

centrifugal force) into a known geometric distribution. For example, it would not be possible to use a single radioisotope source in the (central) interior of the tank and an array of radiation detectors on the exterior. If such a system were used to gauge propellant that may assume a random distribution (as in zero-g), a large error in propellant quantity measurement would result as is clearly envisioned in one case where the propellant is distributed along the walls of the tank, whereas in another case it is concentrated around the source in the center of the tank. The mean propellant thickness is the same in both cases. Since gamma ray absorption is a function of the thickness of the absorber, there will be equal amounts of gamma radiation absorbed within the propellant in both cases, and equal amounts of radiation will reach the detectors despite the fact that the first case has perhaps nearly an order of magnitude more propellant than the latter case. Needless to say, an attempt to use such a system for gauging the propellant quantity will produce a very large error.

To overcome this problem of random fuel orientation, the sources and the detectors are configured into arrays and distributed on a (nearly) planar surface, thereby creating a uniform gamma ray field within the boundary of the tank. In this gamma ray field, an element of propellant may be moved anywhere within the boundary of the tank and, regardless of its position, will absorb the same amount of gamma radiation.

Due to limitations on the number (and distribution) of sources and detectors that can practicably be employed, nucleonic gauging invariably depends upon discrete sampling of the tank interior; a typical configuration is illustrated in Figure 2-1.

As shown, the gauge consists of individual gamma ray radioisotope sources, each of which is external to the tank and collimated to produce radiation in a narrow cone. On the opposite side and also external to the tank, a single detector is positioned in the center of each of the cones of radiation. Each source is sufficiently collimated so that it essentially illuminates only its opposing detector. The combination of each source and its opposing detector comprise a "source-detector pair", thereby providing a discrete "sample column" through the interior of the tank. By configuring a number of such source-detector pairs into an appropriate "sampling array", the entire contents of the tank can be sampled.

In general the requirements for a radioisotope source for propellant measurement are high energy penetration capability to permit source-detector installation external to the tank, a comparatively long half-life to minimize calibration, and a source strength consistent with the accuracy and resolution requirements without compromise to personal safety.

2.1 PRINCIPLE OF OPERATION

In the nucleonic gauging system, the mass sampled by the $i^{\underline{th}}$ sample column due to the exponential attenuation of primary radiation from a

radioisotope source when an absorber and/or fuel intervenes the gamma ray beam will be proportional to the number of photons received at the i^{th} detector, N_{R_i}, according to:

$$N_{R_i} = N_{o_i} e^{-\left(\frac{\mu}{\rho}\right)\rho\ Z_i} \tag{2-1}$$

where

N_o = number of counts received in the absence of an intervening mass

$\frac{\mu}{\rho}$ = mass absorption coefficient

ρ = density of the intervening mass

Z = thickness of the intervening mass

In the event of a tank containing both fuel (liquid) and gas (ullage), the number of counts received by the i^{th} detector is simply

$$N_{R_i} = N_{o_i} e^{-\left(\frac{\mu}{\rho}\right)\left[\rho_L\ Z_{L_i} + \rho_G\ Z_{G_i}\right]} \tag{2-2}$$

where the subscripts L and G refer to the liquid and gas, respectively, within the i^{th} sample column.

The mass, M_i, in the i^{th} column is given by

$$M_i = (W_i)\ A_i \left(\rho_L Z_{L_i} + \rho_G\ Z_{G_i}\right) \tag{2-3}$$

$$= (W_i)\ A_i \left(\frac{\mu}{\rho}\right)^{-1} \ln\left(\frac{N_{o_i}}{N_{R_i}}\right) \tag{2-4}$$

where A_i is the area associated with the i^{th} sample column and W_i is the weighting factor associated with the i^{th} source-detector pair. In the typical situation where the sample columns are parallel

$$A_i \approx \frac{A_T}{k} \text{ and } W_i \approx 1 \text{ for all } i \tag{2-5}$$

where A_T is the total projected cross-sectional area of the tank and k is the number of sampling columns.

Thus, the total mass sampled by the k sample columns is

$$M = \sum M_i = \frac{A_T}{k} \sum_{i=1}^{k} \rho_L Z_{L_i} + \rho_G Z_{G_i} \tag{2-6}$$

or:

$$M = \frac{A_T}{k}\left(\frac{\mu}{\rho}\right)^{-1} \sum_{i=1}^{k} \ln \left(\frac{N_{o_i}}{N_{R_i}}\right) \tag{2-7}$$

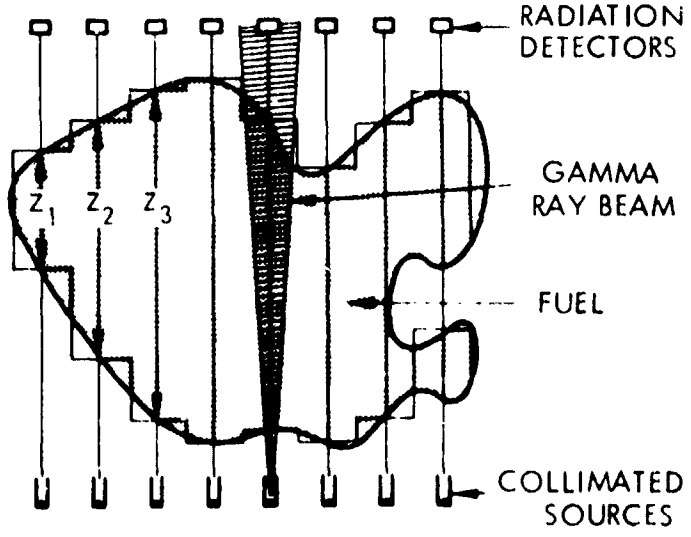


Figure 2-2. Two-Dimensional Area-Weighting for Computing Propellant Quantity

This technique for computing propellant quantity is graphically illustrated (in two dimensions) in Figure 2-2.

It is now noted that the number of sample columns are finite and, hence, the "calculated" mass will not (necessarily) be precisely the "actual" mass. Also, the calculated mass will depend upon the configuration (spatial distribution) of the sample columns and the tank size and geometry. What is required, then, is a conversion – or, more correctly – a calibration curve from which the actual mass can be determined from the calculated mass, i.e.,

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

$$M_{act} = .f(M_{calc}) \qquad (2\text{-}8)$$

In practice, Equation (2-8) is solved with a computer simulation wherein the propellant can assume random orientations within a propellant tank under zero-g conditions. (See Reference 1.) However, Equation (2-7) is very easily mechanized with a small capacity digital computer and, a priori, so too are the functions represented by Equation (2-8) for one-g conditions.

SECTION I.

GAUGING SOFTWARE DESCRIPTION

A major consideration for any flight article is the minimization of both size and weight. Therefore, considerations for a system controller computer were in the field of mini- or micro-computers. A Control Data Corporation, model 469 mini-computer was chosen because of its high reliability in applications where small size, light weight, and low power requirements are a must. (A complete description of the computer can be found in Section 6.)

The software used for the gauging system is referred to as the 469 assembly language. This language is similar in application to any assembly language but unique to the 469 computer. A complete explanation of the language is presented in References 2, 3 and 4.

Figure 3-1 presents a flow diagram of the overall program structure. The program can easily be separated into three operating modes, each independent of the other except for data linkage. These three modes are: Program Initialization, Mass Calculation, and Data Verification.

3.1 PROGRAM INITIALIZATION

The computer used for this system is a CDC 469 mini-computer with $20,000_8$ words of memory. Of this memory, the first 2000_8 words are defined by hardware as the DRO (destructive readout) region commonly referred to as the scratch pad. Of this 2000_8 words of scratch, the first 400_8 words are referred to as page zero, and the first 20_8 locations of page zero are the processor registers. Refer to Figure 3-2 for a graphic description of the memory allocations.

The DRO region has three characteristics that play an important part in the development of the program. First, this region may be destroyed randomly at power-up and power-down or with the depressing of the master clear button. This makes it impractical for program storage. Since hardware defines a starting location of 400_8, the first function of the software is to transfer control to the NDRO region (2000_8), which is preserved during these critical times. Secondly, the DRO region can be written into during program execution. (The NDRO region cannot be written into without first depressing the NDRO button on the programmer's console. This prevents inadvertently writing over the original program during execution.) Therefore, the DRO region is used for temporary data storage. Thirdly, page zero has the unique ability to be addressed directly for read and write instructions; other pages must be referred to by indexed or indirect addressing. This ability to read or write with direct addressing saves both execution time and program storage. Page zero is best used for storage of any constants that are used frequently.

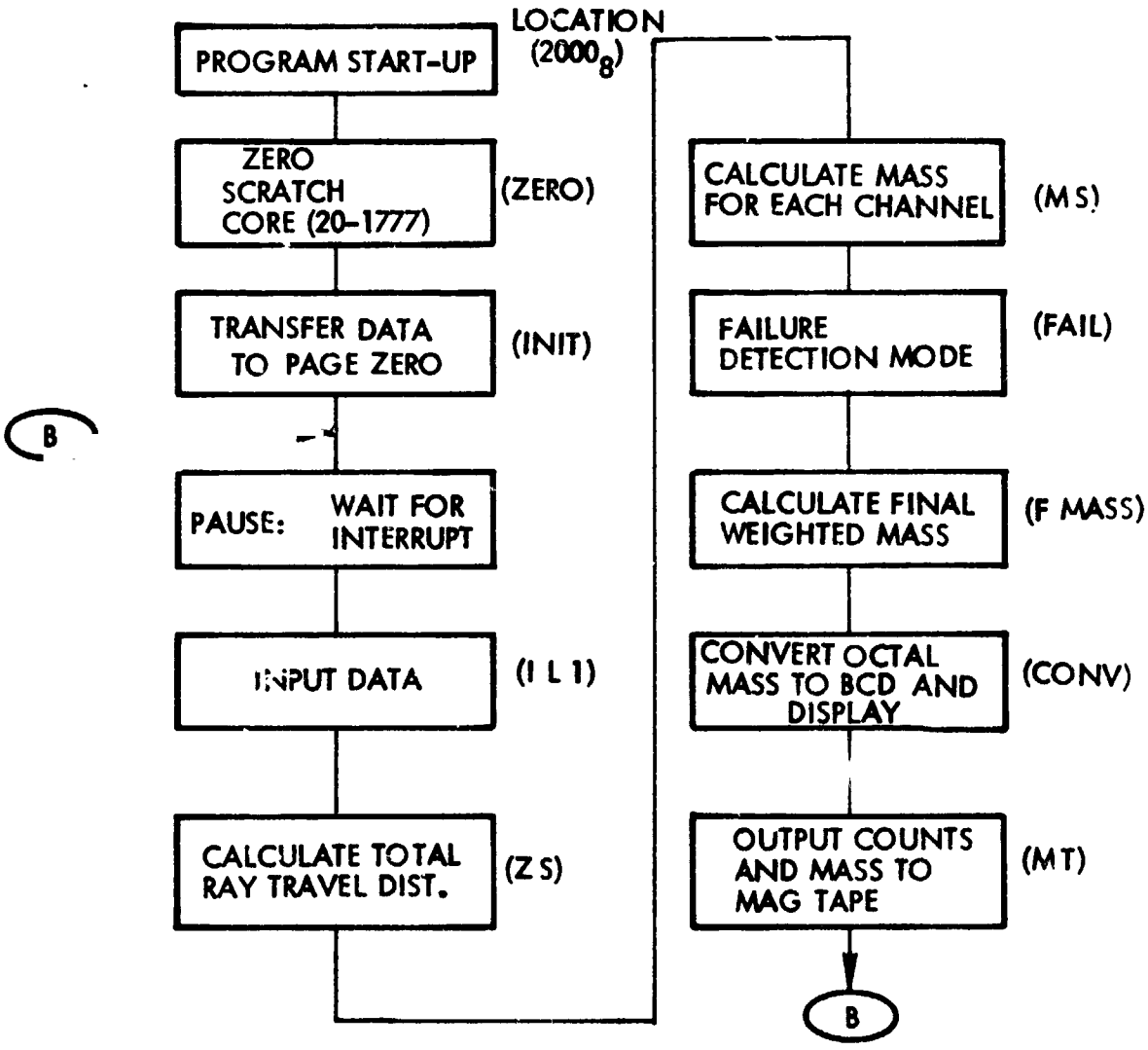


Figure 3-1. Program Startup

The first function of the software is to transfer control to 2000_8 (the start of the NDRO region); the second function is to zero the scratch pad from one location 20_8 to 1777_8. All the frequently used constants can now be transferred to page zero.

As mentioned previously, hardware defines a starting address of 400_8. At power-up, the P register (program counter) will therefore display 400_8. Initially, control is transferred to START (2000_8) by an unconditional jump (UJ), which is stored at core locations 400_8 and 401_8. Since this instruction is in the DRO region, it may be inadvertently destroyed; it is rewritten to lower core from the write protected region at each program startup.

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

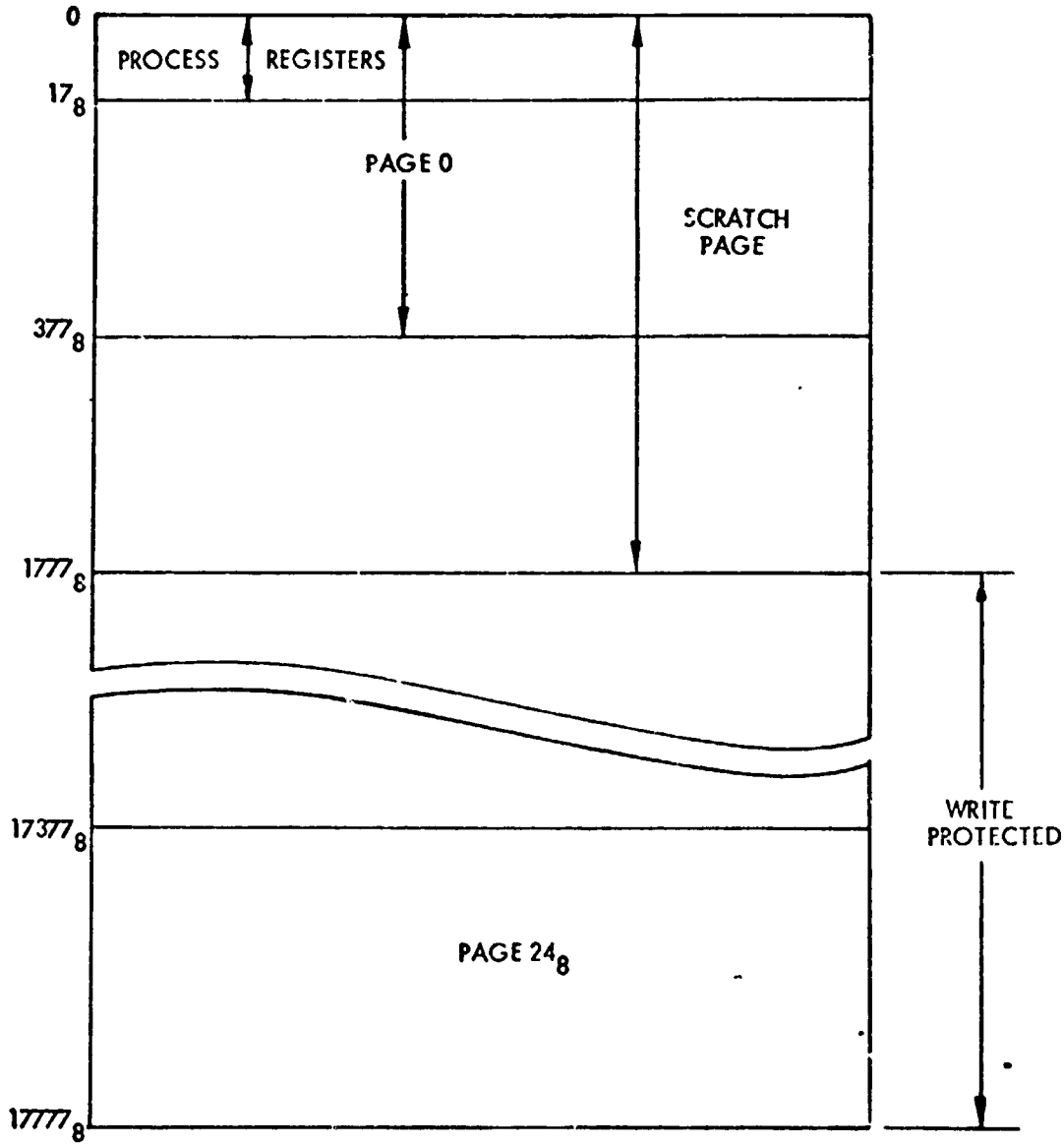


Figure 3-2. Core Allocation

Certain options have been incorporated into the software to operate various peripherals. If the data are to be retained for further analysis via either mag-tape or teletype, the corresponding flag must be set according to Table 3-1. The system is constructed to run completely independent of any peripherals, or with the mag tape and/or teletype connected. One other option is the use of the Signal Conditioning Electronics for data input or to use constant data. Using constant data is useful for program checkout, though once the program is functioning properly, the inputs from the SCEs should always be used.

Table 3-1. System Flags

Variable	Core Location	Value	Description
GOMT	2023_8	0*	Perform mag-tape write
		1	Skip mag-tape write
GOTEL	2025_8	0*	Perform teletype write
		1	Skip teletype write
KFLAG	2027_8	0*	Use SCE input
		1	Use constant inputs

*Nominal values

If indeed, the mag-tape is to be used, one other function is to be performed by the initialization portion of the software. The mag-tape is positioned at the beginning of tape and a dummy file is generated. This dummy file is used by the search and read routines for the mag-tape.

Control is now transferred to the teletype, and the flag "Ready" is displayed. At this time any descriptive information may be entered onto the mag tape. Usually at this time, the date, fuel loading, and any particulars of the run are entered. Input is terminated with the $ key, and control is again transferred to the console. The program is now in a pause, waiting for a level three interrupt issued by hardware to acknowledge the beginning of the mass calculation sequence of the code.

3.2 MASS CALCULATIONS

This portion of the program will continue uninterrupted until such time as an input from the console is received. The total propellant mass is basically generated by a polynominal curvefit. The input from the SCE's records the actual number of counts received by the detectors for

any 1-second interval. These count data can then be directly converted to a ray travel distance through the intervening mass by the following equation:

$$Z_i = \frac{1}{\mu} \ln \frac{N_i}{N_{e_i}} \qquad (3\text{-}1)$$

where:

Z_i = ray travel distance through the intervening mass (cm)

μ = propellant absorptivity (cm^{-1})

N_i = counts received via channel i

N_{e_i} = counts received for an empty tank via channel i.

The Z's can then be used to calculate the propellant mass within the tank by means of a curve-fitting subroutine stored in core in the 469. This curve-fitting subroutine is referred to as the calibration curve(s) and is previously determined off-line in a large scale scientific computer. Essentially, a Monte Carlo simulation of the propellant behavior in zero-g conditions for the given tank/gauge geometry is employed to relate the (average) Z's to the actual mass within the tank. This Z-mass relationship is (invariably) in the form of a polynomial, and it is the degree of this polynomial and its coefficients which are stored in the 469 core. Note that this polynomial (the calibration curve) is determined uniquely for each particular tank/propellant/gauge-configuration.

For each source/detector pair there is both a Z and a mass calculation made based on its respective calibration curve. After the masses have been computed, a failure analysis must be performed to check their validity. The four masses (there are four source/detector pairs-channels) are compared against one another, and the differences are recorded. If one mass consistently has a large difference when compared to the others, that channel is considered inoperable, and the respective mass is weighted out of the total mass calculation. The system can operate with the loss of one channel, but the loss of two or more channels will result in the shutdown of the entire system.

In the event of system shutdown the display is blanked, but the system will continue to operate. Inputs are still accepted and masses generated, and the failure analysis still performed. If for any iteration the masses have stabilized sufficiently to pass the failure analysis, the display will monitor this new mass, and the system will begin to operate again. The mass calculations will continue indefinitely until instruction from the console to halt the mass calculation sequence is received.

3.3 DATA VERIFICATION

After each iteration of the mass calculations, the console is checked for an input. At the sign of the first non-zero input the system will discontinue its mass calculations, write an end-of-file mark onto the mag-tape, and pause for further instructions from the console. At this point the operator has two choices: first, he may push the STEP, MASTER CLEAR, and RUN buttons, in that sequence, to start the program from the very beginning, or he may verify that data have been entered and written to the mag-tape. By changing the input from the data register of the console, the program goes into the data verification portion of the software. By data verification, the system verifies that data have been taken, not that the data are correct. To do this, the mag-tape is rewound one file, and the data acquired during the last phase are displayed on the teletype. Six columns of information are displayed, all in octal. The first four are the count data accepted from the four source-detector pairs. This information will enable the operator to see if the four channels are operating correctly. The fifth column will be the total mass of the system generated by the zero-g gauging system. The sixth column will be the total mass generated by the Rho-V gauging system (not implemented at this time). Once again these quantities are displayed in octal and must be converted to decimal for comparison with the mass displayed by the system hardware.

After completion of the data dump to the teletype the system will restart itself by displaying "ready" on the teletype. (Note: the system should be allowed to display all the data to insure correct system operation. Early termination will result in the loss of data.) At this point, the program has been re-initialized, and the operator can enter new run information and start the mass calculations again.

3.4 CHANNEL INPUT DATA FORMAT

The data read from the accumulators must be made available to the 469 computer in a very specific manner. The total counts received by each individual accumulator may range from 100 counts (a full tank) to a maximum of 2 million counts (an empty tank). Since a normal, single-precision data word represents 15 bits of information, a maximum data value of $2^{16} - 1$ (65535) implies double precision inputs are necessary. Therefore, two words of information, and thus two input channels, must be used for each accumulator.

Double precision is correct only in the sense that two data words are used for input. In reality, a modified word organization is utilized. The entire second word is not needed, and status information is inputted as an additional 4-bit piece of information as shown in Figure 3-3. Notice that the "sign" bit in the second word is used as a data-bit along with the next adjacent five bits. This enables a maximum reading of $2^{21} - 1$ (2, 097, 151) counts to be processed. The next four bits represent a status flag, indicating the operational status of the four external accumulator devices. Modified program execution must be made as a function of the failure of any or all of the accumulators. The last six bits are set to zero and serve no particular function at this time.

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

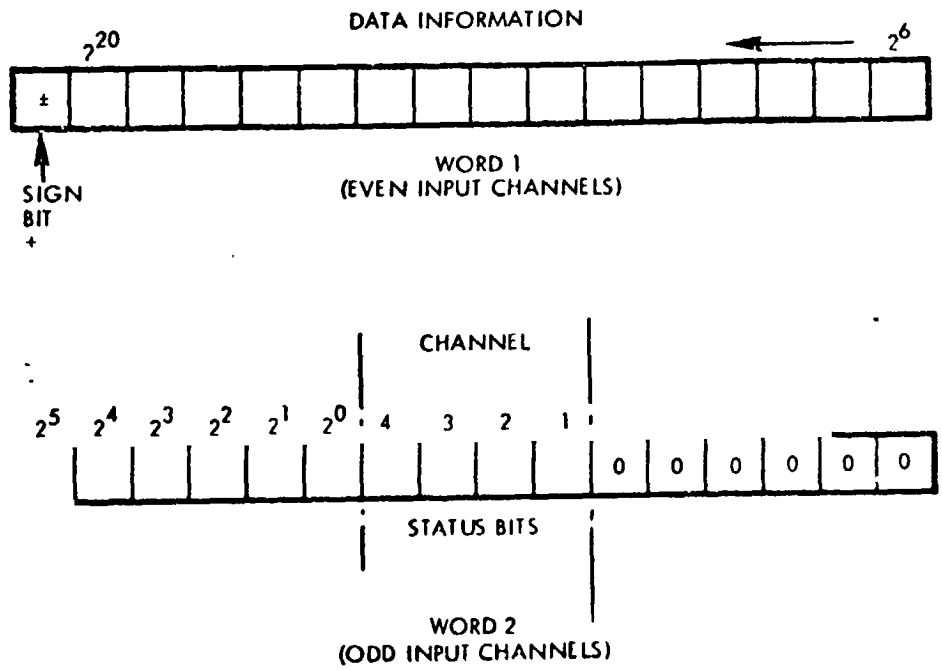


Figure 3-3. Data Word Structure

3.5 SCALING ALGORITHM

There are basically two types of numerical values a variable can attain during the execution of a computer program. The first type, integer values, is generally used where whole numbers are either necessary or convenient. The second type, real values (commonly called floating point numbers), are generally used for all computational values where greater accuracy is needed. On most computer systems the limits for both real and integer values are sufficiently large to alleviate any problem due to overflow; i.e., to generate a number larger than the machine can comprehend. On the CDC 469 mini-computer the limit for integer values is sufficiently large, but real numbers must adhere to the following constraints:

$$0 \leq \text{Real no.} < 1.$$

Obviously, many computations will not satisfy this constraint, and, therefore, scaling each number becomes a necessary function.

To ease the burden of scaling, every real number is represented in core as two words. The first word represents a floating point number

Table 3-2. Data Storage Allocations

Core Location (Octal)	S/D1		S/D2		S/D3		S/D4	
201	Input Data Channel 6 and 7 (2 Words)		Input Data Channel 10 and 11 (2 Words)		Input Data Channel 12 and 13 (2 Words)		Input Data Channel 14 and 15 (2 Words)	
211	Scaled Data Input	Scale Factor	Scaled Data Input	Scale Factor	Scaled Data Input	Scale Factor	Scaled Data Input	Scale Factor
221	$\frac{N_1}{N_0}$	Scale Factor	$\frac{N_2}{N_0}$	Scale Factor	$\frac{N_3}{N_0}$	Scale Factor	$\frac{N_4}{N_0}$	Scale Factor
231	$\ln\left(\frac{N_1}{N_0}\right)$	Scale Factor	$\ln\left(\frac{N_2}{N_0}\right)$	Scale Factor	$\ln\left(\frac{N_3}{N_0}\right)$	Scale Factor	$\ln\left(\frac{N_4}{N_0}\right)$	Scale Factor
241	Z_1	Scale Factor	Z_2	Scale Factor	Z_3	Scale Factor	Z_4	Scale Factor
251	M_1	Scale Factor	M_2	Scale Factor	M_3	Scale Factor	M_4	Scale Factor
261	$\frac{dM_1}{dZ_1}$	Scale Factor	$\frac{dM_2}{dZ_2}$	Scale Factor	$\frac{dM_3}{dZ_3}$	Scale Factor	$\frac{dM_4}{dZ_4}$	Scale Factor
271	ΔM_1	Scale Factor	ΔM_2	Scale Factor	ΔM_3	Scale Factor	ΔM_4	Scale Factor
301	$(N_1)^{1/2}$	Scale Factor	$(N_2)^{1/2}$	Scale Factor	$(N_3)^{1/2}$	Scale Factor	$(N_4)^{1/2}$	Scale Factor
321	Octal	Mass						
322	Scale	Factor						
323	Octal	Integer	Mass					
324	BCD	Mass						

Nomenclature:

N_i = Counts received per channel i

N_o = Maximum counts received (empty tank)

Z_i = Total ray travel distance (cm)

M_i = Mass per channel i (gm)

$\frac{dM_1}{dZ_1}$ = The percent change of the mass with respect to the z length

ΔM_i = Mass uncertainty (gm)

satisfying the previous constraint. The second word represents a scale factor by which Word 1 is scaled. To obtain the true value for a particular variable, the following operations must be performed:

1) Evaluate Word 1 by real conversion from octal to decimal. Note: the decimal place is implicitly placed to the right of the left-most digit; the first digit (left most) represents either negative or positive numbers (0 = positive, 1 = negative).

Core Representation	Octal Value	Decimal Value
040000	0.40000	0.50000
060000	0.60000	0.75000
120000*	-0.60000	-0.75000

2) Evaluate Word 2 by integer conversion from octal to decimal.

Core Representation	Octal Value	Decimal Value
000003	3	3
000010	10	8
177773*	-5	-5
177432*	-346	-230

3) Evaluate the real value of Word 1 and Word 2 by the following algorithm:

$$X = (\text{Word 1}) \times 2^{-(\text{Word 2})}$$

(See Table 3-3 for further example.)

Tables 3-3. Conversion Examples

	Core Representation	Octal Value	Decimal Value	Total Value
Word One	060000	0.60000	0.75	12.00
Word Two	177774	-4	-4	
Word One	130000	-0.50000	-0.625	-640.00
Word Two	177766	-12	-10	

*Negative numbers are represented in core by their "2's" complement.

All real values represented by two words can be evaluated in this manner. The only exception is the natural logarithm computation. Its scaling system is described in the applicable logarithm subroutine description, Section 4.1.

One of the inherent problems with real number computation is the lack of significant figures associated with single precision operations. Since single precision offers only five significant figures (a function of the CDC 469's 16-bit word) it becomes necessary to rescale at a fairly frequent interval to assure a maximum number of significant figures. The scale factor (Word 2) is actually the number of left or right shifts performed to obtain this maximum number of significant figures. A more complete description of the scaling process can be found in the SCALE subroutine description in Section 4.

SECTION IV

AUXILIARY SUBROUTINES

The major portion of the calculations are performed by various auxiliary subroutines. The driver's or main program's sole purpose is to control the interaction between these auxiliary subroutines. Each subroutine performs a specific task or calculation based on a specific set of inputs. This set of inputs, as well as the output, are generally transferred to and from the main driver via the registers. The following section describes the functions of each subroutine, the registers used, and shows a simplified block diagram.

4.1 INIT (Shown in Figure 4-1)

The CDC 469 computer is a $20,000_8$ word machine with the majority of its core being write-protected (non-volatile). It is desirable to have the program residing in this area since it will not be accidentally destroyed by either a random write or loss of power. The only section that is not write-protected is the scratch pad, words 20_8 to 2000_8. This scratch pad, as the name implies, is used for temporary storage of diagnostic information and calculations.

The first page of scratch, words 20_8 to 377_8, have the added capability of being addressed directly. This is a very fast method of retrieving information directly from core. For this reason, any frequently used constants are stored here to enhance the speed of the program. Since the scratch could be destroyed randomly or by power failure, it is not wise to store these constants there permanently. Therefore, any constants that are to be frequently used via scratch, are first loaded into core in the higher write-protected areas, then at program startup, they are transferred to page zero. This is the purpose of INIT. One now has the advantage of permanently stored constants in the write-protected memory, as well as the added advantage of direct addressing of these constants via page zero.

Calling Sequence:

```
SRJ     INIT, 17
```

Registers Used:

R2 = Location of data (page zero)

R3 = Location of data (write-protected memory)

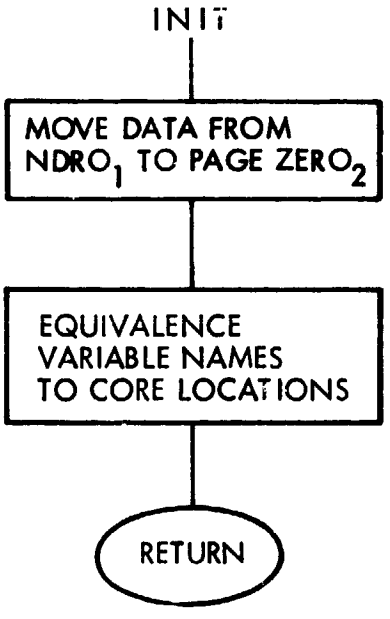
R4 = Temporary

R5 = Loop Counter

R17 = Exit

Required Routines:

None



CORE LOCATIONS 2000_8 TO 20000_8 ARE WRITE PROTECTED

Figure 4-1. INIT

4.2 ZERO (Shown in Figure 4-2)

The purpose of the routine is to "zero" the scratch pad (location 20_8 to 1777_8). Since the computer is designed by hardware to start execution at location 400_8, and the program resides at 2000_8, a subroutine jump (SRJ) must be re-stored at this location after core has been zeroed. This will immediately transfer control to the first word of the program.

Core Location (Octal)	Contents (Octal)	Description
400	400	SRJ – Subroutine Jump
401	2000	Transfer address

Calling Sequence:

```
SRJ     ZERO, 17
```

Registers Used:

R1 = 00000

R2 = Memory scratch address (17_8 to 2000_8)

ORIGINAL PAGE IS POOR

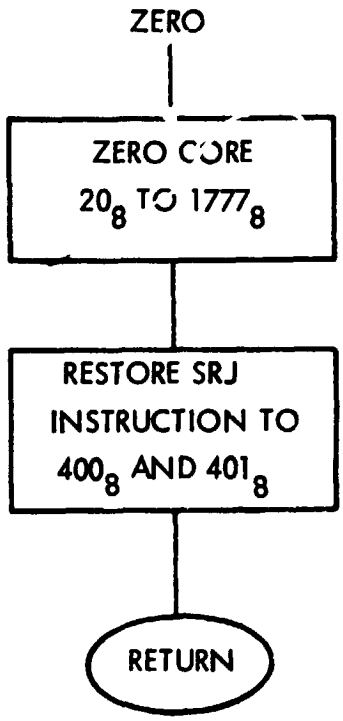


Figure 4-2. ZERO

R3 = 1777_8 (2000_8-1)

R17 = Exit

Required Routines:

None

4.3 IL1 (Shown in Figure 4-3)

At 1-second intervals during the execution of the program, a hardware generated interrupt alerts the computer that data are available on the eight input channels. Once the data are available it is immediately read into memory by this routine.

The data inputs are simple binary integers. A single 16 bit word has a maximum integer value of 77777_8. The data inputs may range from 0 to $1,000,000_{10}$ counts. This presents a slight problem since the value of the counts received is greater than the capacity of a single word. Therefore, two words are used for data inputs. The data format is described in Section 3.4. For calculation purposes, this integer value for each channel is converted to a simple floating point number, referred to as the scaled value. A complete description of the scaling algorithm is given in Section 3.5. The purpose of this routine is then to receive the counts from each of the four channels, convert them from two integer words to a double precision floating value, and store for further use.

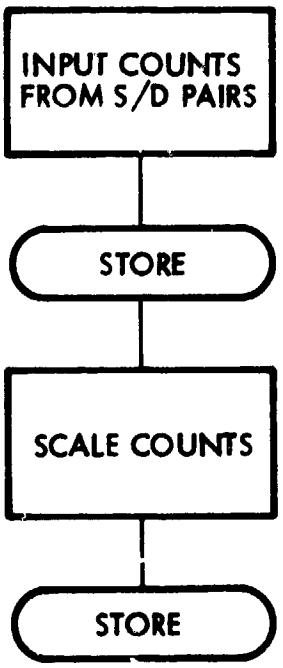


Figure 4-3. IL1

Registers Used:

R6 = Temporary
R12 = Input Word 1
R13 = Input Word 2
R14 = Temporary
R15 = Masking element
R16 = Exit

Calling Sequence:

Since IL1 is initiated upon a hardware interrupt, no calling sequence is available.

Required Routines:

SCALE

4.4 ZS (Shown in Figure 4-4)

This routine generates the actual ray travel distance through the fuel in centimeters. These calculations are based on the number of counts received by the detectors under empty tank conditions and a gamma absorption coefficient. Since the standard deviation of the count rate is essentially the square root of the counts received, it is conceivable the

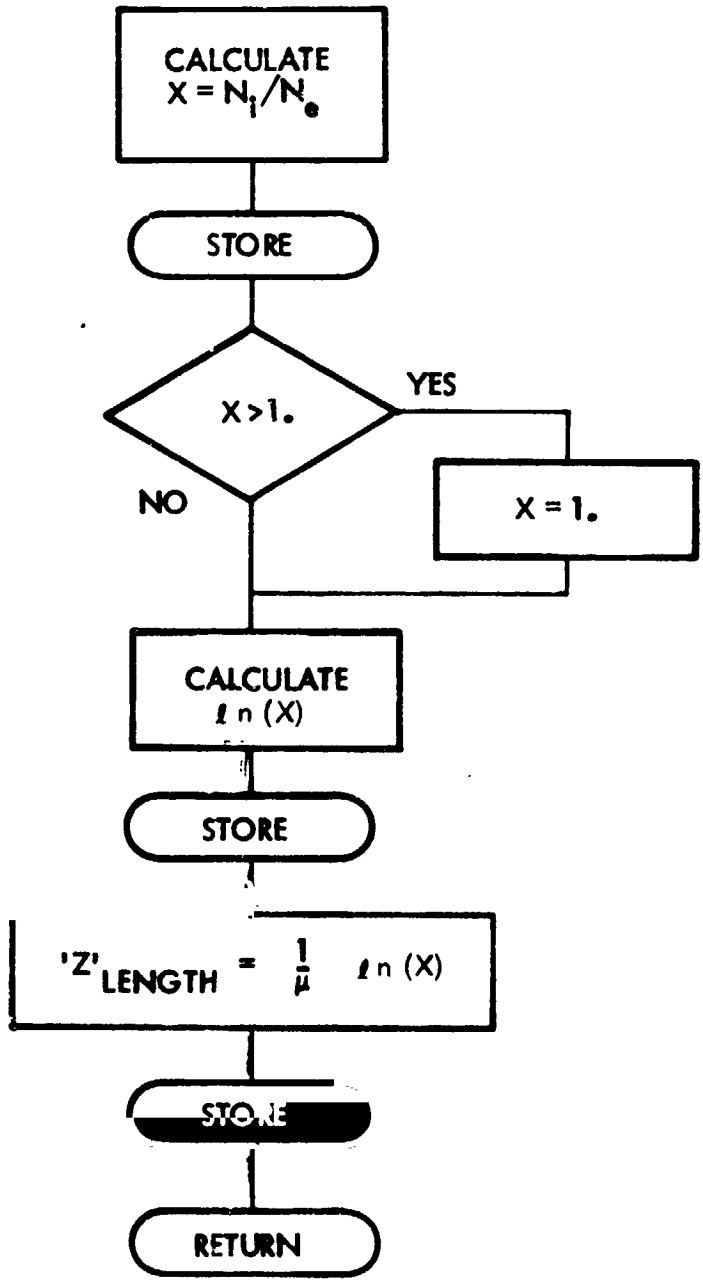

CALCULATE
X = N_i/N_e
STORE
X>1.
YES
NO
X = 1.
CALCULATE
ℓ n (X)
STORE
'Z' LENGTH = 1/μ ℓ n (X)
STORE
RETURN

Figure 4-4. ZS

counts at low mass readings will exceed the counts received at empty tank calibrations. In this situation the counts received are set equal to the calibration inputs. All calculations in this routine are based on the scaled inputs as is the evaluation of the logorithm. (See Section 5.)

Algorithm:

$$Z = \frac{-1}{\mu} \ln \left(\frac{N_i}{N_e} \right)$$

where

μ = absorption coefficient (cm^{-1})

N_i = counts received

N_e = counts received with an empty tank

Z = gamma ray travel distance through the fuel (cm)

Calling Sequence:

```
SRJ       ZS, 17
```

Registers Used:

R2 = Location of data
R3 = Location of N_e
R6 = Counts/answer out
R7 = Scale factor
R10 = N_e
R11 = Scale factor
R12 = Temporary
R15 = Loop counter
R17 = Exit

Required Routines:

ADDR, DIV, LOG, SCALE

4.5 MS (Shown in Figure 4-5)

This routine performs the actual mass calculations for each of the four channels as a function of their respective "Z" length (the gamma ray travel distance through the propellant). This computation is actually

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

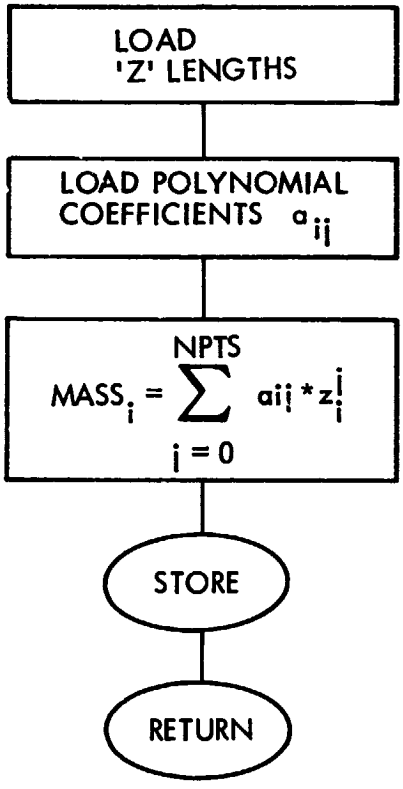


Figure 4-5. MS

a polynomial evaluation based on prior data accumulation and polynomial curvefit. A representative sample size of mass versus Z length for each channel is gathered and fed into an offline computer system with a curvefit routine. Each channel's data may be subdivided into two groups. For example, a linear curvefit may suffice for Z lengths greater than 20 centimeters, while a third degree polynomial might be required for Z lengths of less than 20 centimeters. A polynomial curvefit is then performed for each of these data samples. The result is a set of coefficients for each polynomial that represent the curve(s) for each channel. These coefficients are then fed into the 469 for the mass calculations performed by this routine.

The polynomial curvefit may attain any degree, although because of the word size limitations of the computer, a degree of greater than 4 would achieve no greater accuracy. Polynomials from one segment or channel to the next, need not be of the same degree; the only requirement is that the variable NPTS be set to the largest degree of any of the polynomials.

Calling Sequence:

```
SRJ       MS, 17
```

Registers Used:

R2 = Location of "Z" data
R3 = Location of the coefficients
R4 = NPTS – number of coefficients
R5 = Counter
R6 = Temporary
R7 = Temporary
R10 = Temporary
R11 = Temporary
R12 = Octal mass
R13 = Scale factor for mass
R14 = Z
R15 = Scale factor for Z
R16 = Exit for scaling routine
R17 = Exit

Required Routines:

PICK, SCALE

4.6 PICK (Shown in Figure 4-6)

PICK is used exclusively for the purpose of choosing the proper curvefit for the mass calculations in subroutine MS. This evaluation is based on the variable BKPT. If the calculated ray travel distance, Z, is less than BKPT, the first curvefit for the particular channel is used; if Z is larger or equal to BKPT, the second curvefit is used.

Calling Sequence:

```
SRJ       PICK, 16
```

Registers Used:

R2 = Location of data
R3 = Location of coefficients

R4 = Counter

R5 = Mass index

R6 = "Z" length

R7 = Scale factor for Z

R10 = Temporary

R11 = Temporary

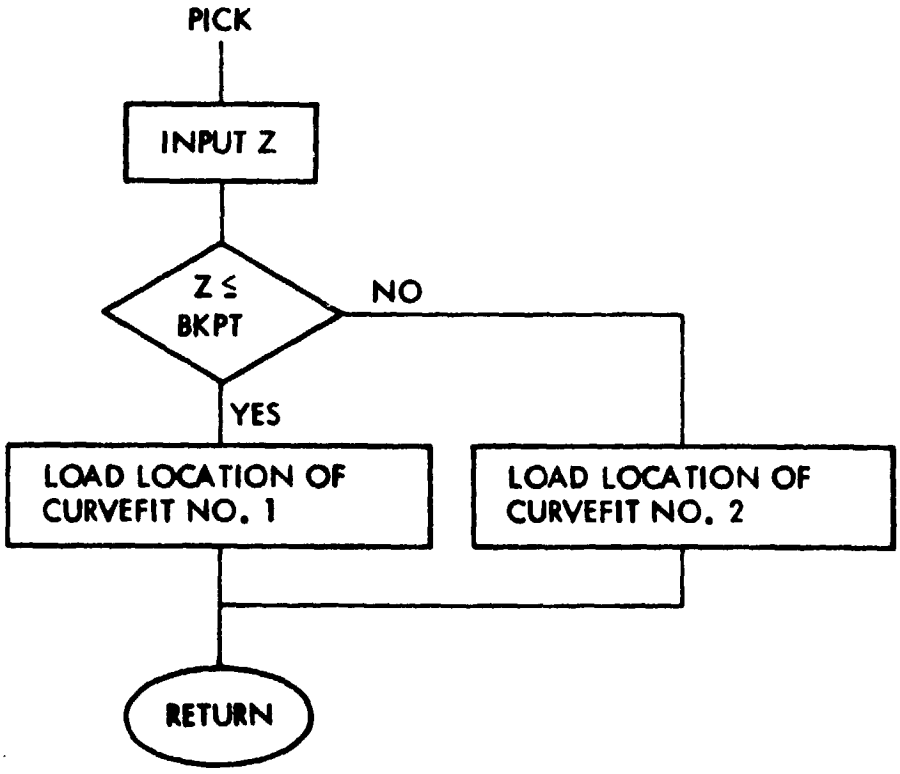
R12 = Temporary

R13 = PLUS

R14 = Z

R15 = Z scale factor

R16 = Exit



NOTE: THE MASS VS 'Z' LENGTH MAY BE DIVIDED INTO TWO CURVEFITS; ZBPT IS T' ONE POINT COMMON TO BOTH CURVEFITS.

Figure 4-6. PICK

Required Routines:

ADDR

4.7 FAIL (Shown in Figure 4-7)

During the execution of the program it is imperative that the four channels are generating "feasible" masses. If all four channels are operating smoothly and their respective masses are comparatively equal, there obviously is no problem. But if a malfunction does arise, this routine must decipher which channels are operating properly and what course of action is to be taken. The three options listed below are available when operating with four source/detector pairs:

1) All channels operating normally

2) One channel failure – three normal channels

3) Two or more inoperative channels.

The system can function properly with three of the four channels available with correct data by eliminating the bad data channel from the final mass calculations. This is done simply by assigning a weighting factor of zero to this particular channel and a weighting factor of 0.3333 to the remaining three channels. (Under normal operating conditions a weighting factor of 0.25 is assigned to each channel.) If two or more channels are considered inoperable, the system will shut down until at least three channels respond to a consistent mass reading.

The following differences are calculated for each 1-second interval:

$$ABS\ (M_1 - M_2) \quad (1)$$

$$ABS\ (M_1 - M_3) \quad (2)$$

$$ABS\ (M_1 - M_4) \quad (3)$$

$$ABS\ (M_2 - M_3) \quad (4)$$

$$ABS\ (M_2 - M_4) \quad (5)$$

$$ABS\ (M_3 - M_4) \quad (6)$$

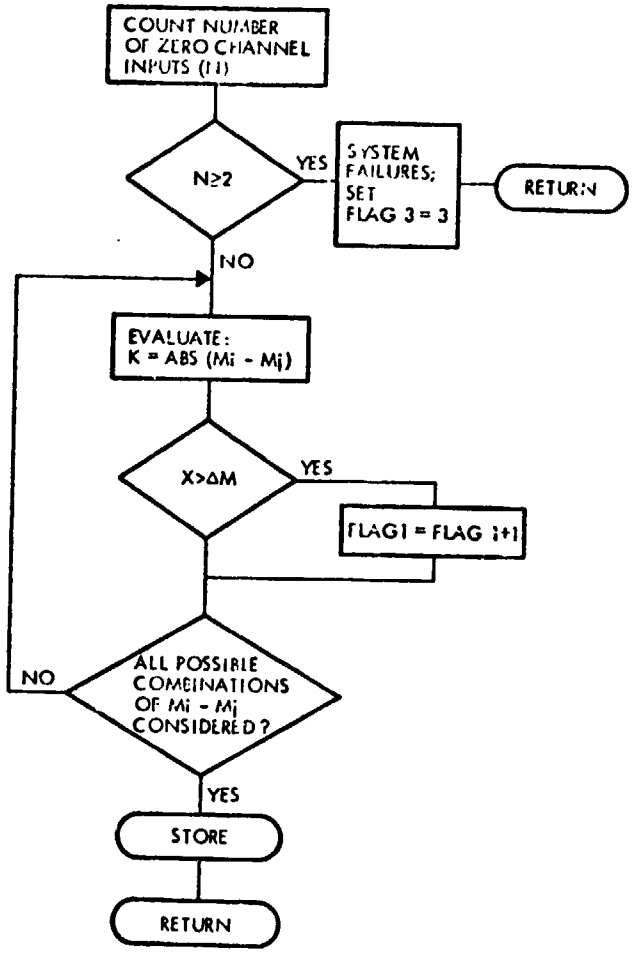


Figure 4-7. FAIL

where:

ABS = ABSOLUTE VALUE

M_i = Computed mass for channel i.

These six quantities are then compared to an error criterion, DELM (currently 72 grams), and the corresponding equation number is accumulated in a register if the absolute value of the difference is greater than DELM. This accumulation of the equation number then will identify the channel in error. For example, if all six quantities are less than DELM, the sum of the equation numbers is zero, signifying all channels are operating properly. If channel three is malfunctioning, then Equations (2), (4), and (6) will fail the comparison test and the sum of the numbers will be 12_{10}. This sum is then stored into the variable FLAG 1 and passed onto subroutine FMASS. One other error indicator is set at this time – FLAG 3. This variable represents the actual number of equations that failed the comparison criteria. This is also passed onto FMASS.

Calling Sequence:

SRJ FAIL, 17

Registers Used:

R2 = Location of $2\Delta M$
R3 = Location of M_i
R4 = Location of M_j
R5 = Temporary
R6 = M_i
R7 = Scale factor for M_i
R10 = M_j
R11 = Scale factor for M_j
R12 = Temporary
R13 = Temporary
R14 = Loop counter
R15 = Loop counter
R16 = Failure Counter
R17 = Exit

Required Routines:

ADDR

4.8 FMASS (Shown in Figure 4-8)

This routine generates the final mass as a function of the four channel mass calculations. Depending on the feasibility of the four input masses determined by FAIL, this routine weights the masses accordingly and sums the results to yield a total system mass. The bulk of the channel failure selection was done in FAIL and only the two error indicating variables, FLAG 1 and FLAG 3, are passed onto the routine. (FLAG 1 represents the sum of the equation numbers of the failed comparison tests, and FLAG 3 represents the actual number of comparison test failures. The following table represents the possible conditions and values of FLAG 1 and FLAG 3.

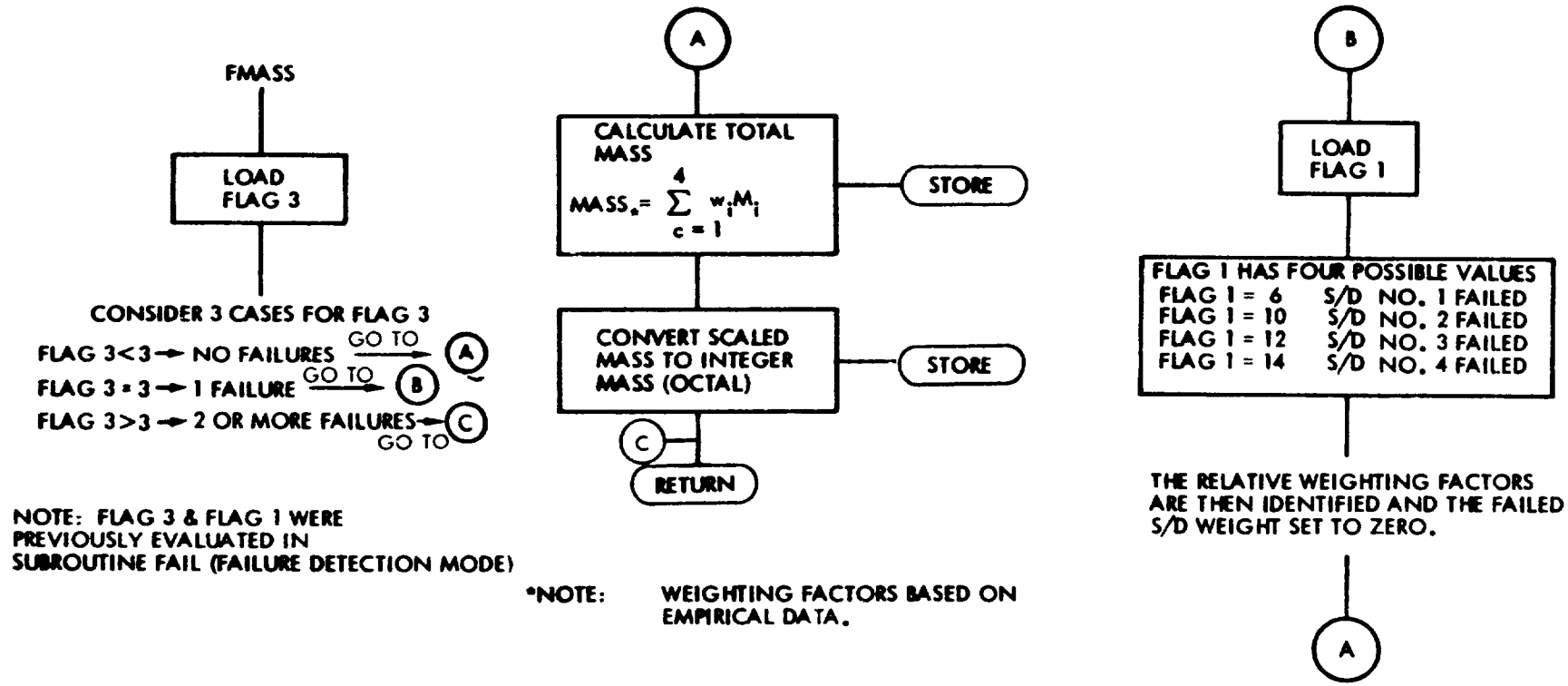


Figure 4-8. FMASS

Condition	FLAG 1*	FLAG 3*
No Channel Failures	0	0
Channel 1 Failure	6	3
Channel 2 Failure	10	3
Channel 3 Failure	12	3
Channel 4 Failure	14	3
2 or more Channel Failures	≥15	≥5

*Octal values

The first function of FMASS is to determine how many channels are operating properly. FLAG 3 is used for this purpose. FLAG 3 equal to zero indicates all channels are operating properly, and the final mass is generated from all four channels by weighting them appropriately ($wt_i = 0.250$). FLAG 3 greater than three indicates two or more channels are inoperable, and the system cannot generate a mass with sufficient confidence. The display is immediately blanked, and the program recycles itself until at least three of the four channels are functioning properly.

If FLAG 3 has the value of three, a single channel failure is detected, and the specific channel must be identified. FLAG 1 is used for this purpose, and a minimal amount of comparison logic is needed to establish which channel is at fault by the use of the above table. Once the malfunctioning channel is identified, it is simply a matter of assigning the proper weighting factor to each mass to generate a total system mass. For example, assume FLAG 1 = 12; referring to the table, it is found that channel 3 is not operating properly. Therefore, the weighting factor for channel 3 is set equal to zero, and the other three weighting factors set equal to 0.3333. Therefore, the total mass is equal to $M_1 + M_2 + M_4$. From the table, if FLAG 1 = 0, all channels are operating correctly, and the respective weighting factors for the four channels are set equal to 0.25.

The total system mass is a scaled floating point number that is now converted to a more meaningful octal integer constant and stored in core for later octal to decimal conversion.

Calling Sequence:

```
SRJ     FMASS, 17
```

Registers Used:

R1 = Location of final mass

R2 = Location of M_i

R3 = Location of weighting factors (Wt_i)

R4 = Temporary

ORIGINAL PAGE IS POOR

R5 = Counter

R6 = $M_i * Wt_i$

R7 = Scale factor

R10 = Sum of $M_i * Wt_i$

R11 = Scale factor

R12 = Temporary

R13 = Temporary

R14 = Failure flag

R15 = Temporary

R16 = Exit for scale

R17 = Exit

Required Routines:

ADDR, SCALE

4.9 CONV (Shown in Figure 4-9)

This is a simple utility routine used to convert the octal integer mass to a BCD (Binary Coded Decimal) single word output. This can then be transferred directly to the display console indicating an integer decimal mass in grams. The algorithm for conversion is as follows:

ABCD = Octal number to be converted

WXYZ = BCD word (Binary Coded Decimal)

$WXYZ = A*8^3 + B*8^2 + C*8^1 + D*8^0$

Note: For octal representation, three binary digits represent an octal digit. For BCD representation, four binary digits represent a decimal digit. For example,

$$101 = 5_8;\ 0101 = 5_{BCD}$$

Similarly:

$$001\ 001 = 11_8 = 9_{10};\ 1001 = 9_{BCD}$$

Calling Sequence:

```
SRJ     CONV, 17
```

Registers Used:

R2 = Location of M_i

R6 = Temporary

R7 = Temporary

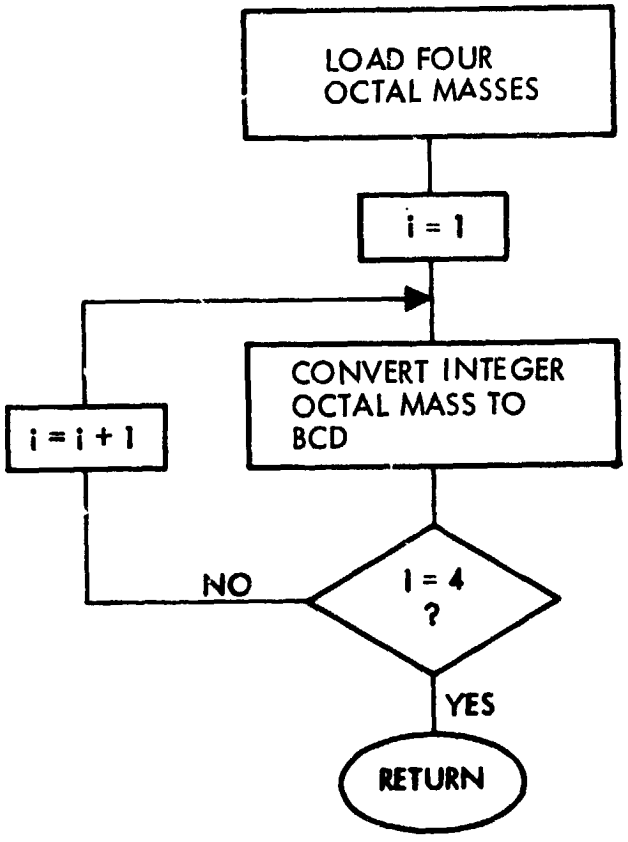


Figure 4-9. CONV

R10 = Number to convert (octal)

R11 = 00000

R12 = Conversion factor

R13 = C0000

R14 = Output – BCD

R15 = Temporary

R16 = Loop counter

R17 = Exit

Required Routines:

None

4.10 DOUT (Shown in Figure 4-10)

DOUT is a utility routine used to display the four count readings and the mass to the teletype in octal notation. Since the system teletype is only capable of printing a line every 2.5 seconds, the routine should not

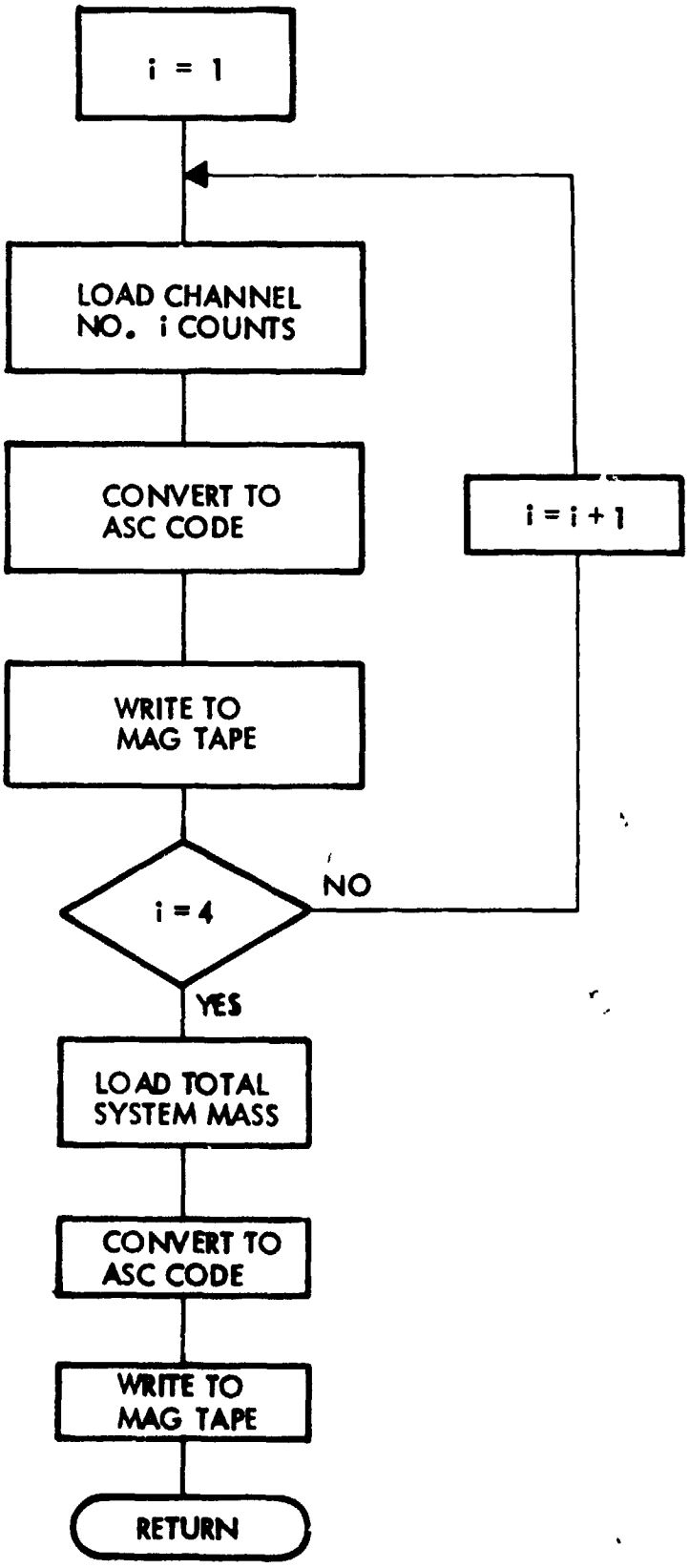


Figure 4-10. DOUT

be called during the operating portion of the gauging system. It should rather be used for tape playback only. The variable GOTEL (Location 2025_8) governs this option as below:

Core Location	Contents	Comment
2025_8	014000	Enables Teletype Output
2025_8	014001	Bypasses Teletype Output

Algorithm:

1) Input character (from Storage)
2) Waits for Teletype not busy
3) Print Character
4) Repeat sequence (number of iterations ≤44).

Calling Sequence:

```
SRJ        DOUT, 17
```

Registers Used:

R1 = Location of data
R2 = Loop counter (4 channels)
R3 = Loop counter (7 characters/channel)
R4 = Masking element
R5 = Temporary storage
R6 = Data register
R7 = Data register
R13 = Zero – ASC code (260B)
R14 = Carriage return (212B)
R15 = Line feed (215B)
R16 = Space (240B)
R17 = Exit register

4.11 LOG (Shown in Figure 4-11)

LOG is a single precision routine written to evaluate the natural logarithm of a scaled floating point input. The input, previously scaled (or rescaled) by routine SCALE, consists of two words. Word 1 represents a floating point number between 0 and 1. Word 2 represents a scale factor (a power of 2). Therefore, X, the floating point input, can be represented by:

$$X = W * 2^K \qquad (4-1)$$

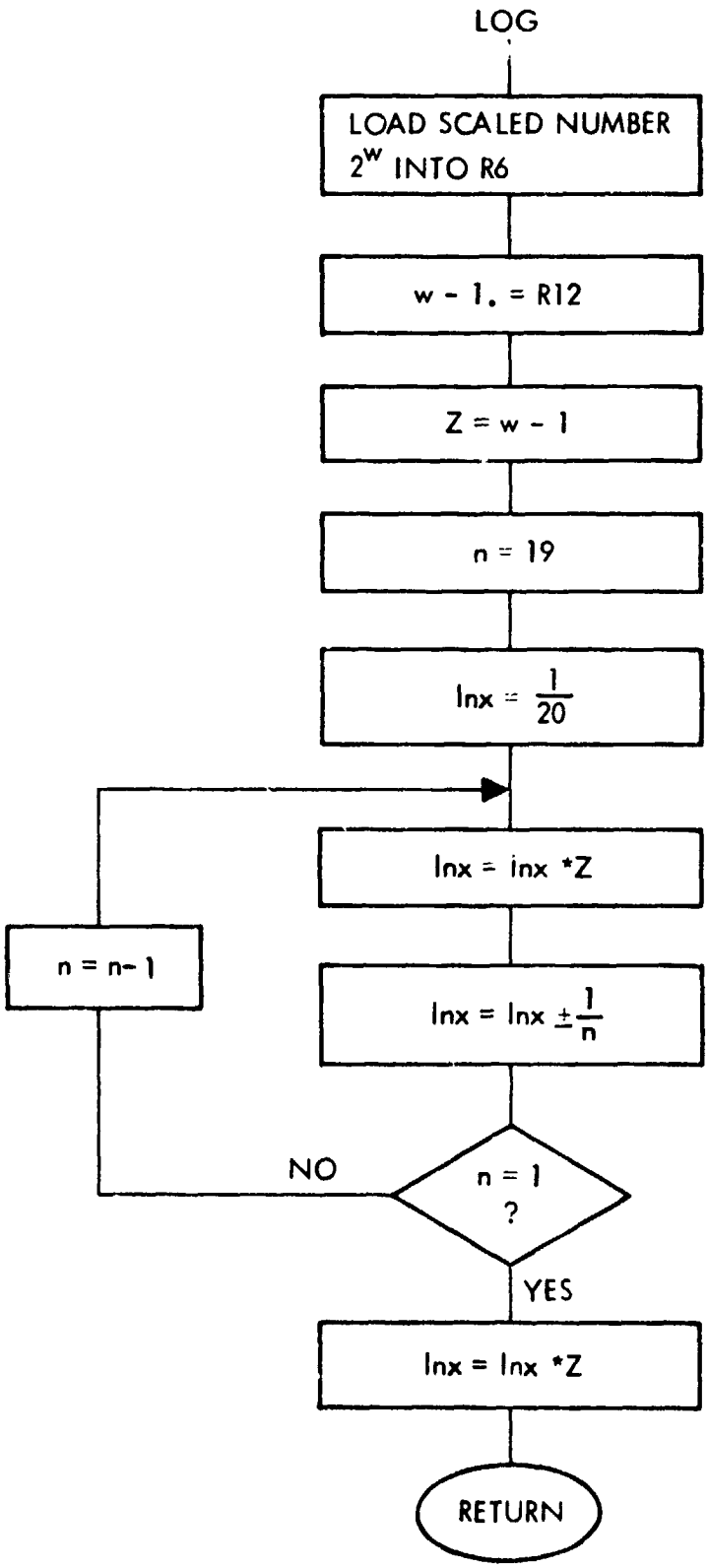


Figure 4-11. LOG

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

where:

W = WORD 1 (MANTISSA) and $0 \leq W < 1$

K = WORD 2 (SCALE FACTOR) and $-\infty < K < \infty$

Since

$$\ln (X) = \ln (W) + K * \ln (2) \tag{4-2}$$

and noting the series expansion for ln (1 + Z):

$$\ln (1 + Z) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1} Z^n}{n} \tag{4-3}$$

for

$|Z| \leq 1$

$Z \neq -1$

and making the substitution, W = 1 + Z, Equation (4-3) becomes

$$\ln (W) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1} (W-1)^n}{n} \tag{4-4}$$

for

$0 < W \leq 2$

Equation (4-4) is a valid expression for ln (W) except at the lower limit where W = 0. However, a simple "zero-check" of the channel inputs causes the series evaluation to be by-passed in the event W = 0 since this condition indicates that counts are not being received (i. e., the particular channel is not operating).

Evaluating Equation (4-4) for a finite number of terms, j, yields:

$$\ln (W) = \sum_{n=1} \frac{(-1)^{n-1} Z^n}{n} + K * 0.69315 + \epsilon \tag{4-5}$$

for

$0 < W \leq 2$

$Z = W - 1$

and where:

$$\ln (2) = 0.69315$$

The quantity, ϵ, in Equation (4-5) represents the error inherent in ln (W) due to only a finite number of terms, j, in the expansion. Obviously, it is desired to keep the value of ϵ small, but consistent with the overall system accuracy. In the nucleonic gauging system, a value of $\epsilon \leq 2^{-15}$ was consistently chosen since it corresponds to the last (15th) bit of a 469 word. In this manner, any additional number of terms in the expansion beyond the minimum number required to yield $\epsilon \leq 2^{-15}$ would not affect the final result.

Algorithm:

let

$$X = W * 2^K \qquad 0 \leq W < 1 \quad \text{(WORD 1)}$$

$$-\infty < K < \infty \quad \text{(WORD 2)}$$

then

$$\ln (X) = \ln (W) + K * \ln (2)$$

noting the series expansion for ln (1 + Z):

$$\ln (1 + Z) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1} Z^n}{n} \qquad \text{if } |Z| \leq 1, \; Z \neq -1$$

let

$$W = 1 + Z$$

then

$$W - 1 = Z$$

substituting yields:

$$\ln (W) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1} (W-1)^n}{n}$$

$$-1 < W-1 \leq 1 \text{ or } 0 < W \leq 2$$

Ln W is found to be a legitimate expression except at the lower limit, i.e., W = 0; at this limit the series is not evaluated since this would imply counts are not being received and that that particular channel is not operating. A simple zero check on the inputs bypasses the series evaluation in this case.

Evaluating ln (W) for n = 20 iterations yields

$$\ln (W) = Z (1 + Z (-1/2 + Z (1/3 + \ldots$$

$$+ Z (1/17 + Z (-1/18 \times Z (1/19 + Z(-1/20)) \ldots)$$

now

$$\ln (2) = 0.69315$$

then

$$K * \ln 2 = K * 0.69315$$

therefore:

$$\ln (W) = \sum_{n=1}^{20} \frac{(-1)^{n-1} Z^n}{n} + K * 0.69315 + \epsilon \quad \begin{matrix} (0 < W \leq 2 \\ Z = W - 1) \end{matrix}$$

and

$$\epsilon = \sum_{n=21}^{\infty} \frac{(-1)^{n-1} Z^n}{n} \leq 2^{-15}$$

Calling Sequence:

```
SRJ     LOG, 17
```

Registers Used:

R4 = Dividend (1.0)
R5 = Loop counter
R6 = W Word 1 input
R7 = K Word 2 input scalefactor
R10 = LOG W Word 1 output
R11 = K Word 2 output
R12 = W-1 Word 1 (double precision)
R13 = W-1 Word 2 (double precision)
R14 = Temporary storage
R15 = Temporary storage
R17 = Exit register

Required Routines:

None

4.12 EXP (Shown in Figure 4-12)

EXP is a single precision routine written to evaluate the exponential of a scaled, two-word input. This routine was written to work in conjunction with the LOG routine. The output of LOG is compatible with the input of EXP. The logarithm input is assumed to be input in the following format:

$$\ln X = \ln W + k \ln 2$$

where

$\ln W$ = Word 1 input

K = Word 2 input

therefore:

$$X = e^{\ln W} * e^{K \ln 2} = W * 2^K$$

Hence, W must be evaluated by the exponential of Word 1 input and then the proper number of right shifts (multiply by 2) performed to obtain the final result. The exponential is found by evaluating a series (see algorithm this section). Sufficient accuracy was obtained by evaluating seven terms of the series.

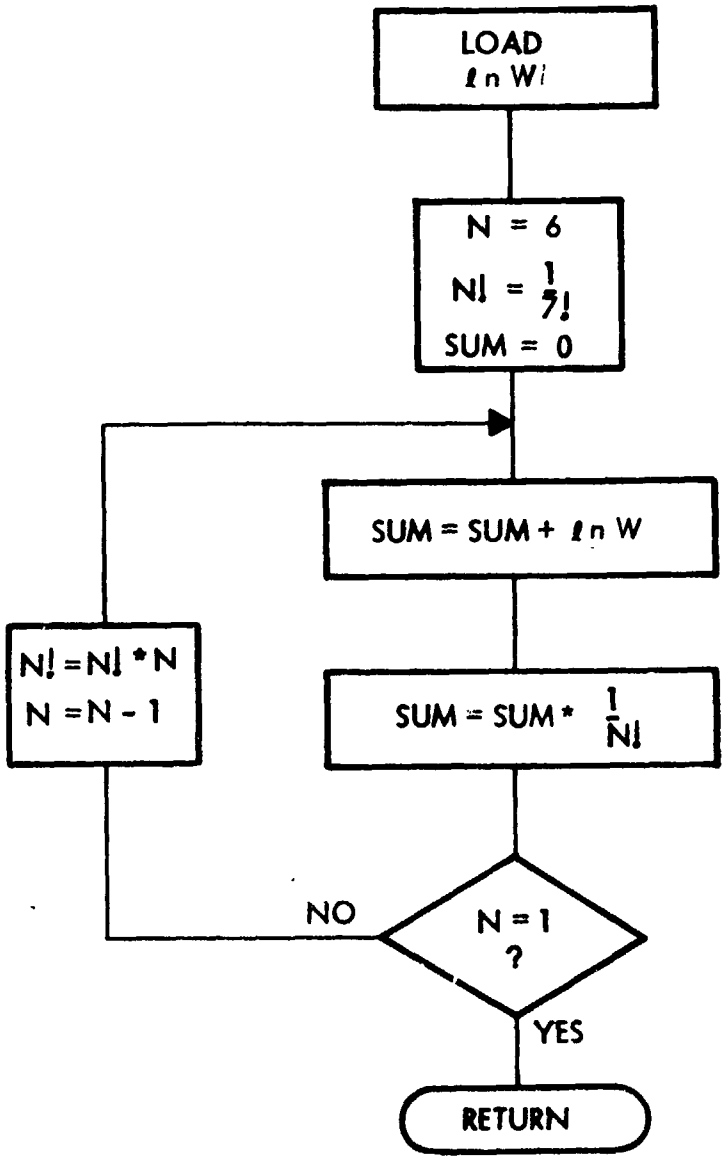


Figure 4-12. EXP

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

Algorithm:

$$e^x = \sum_{n=0}^{\infty} \frac{X^n}{n!}$$

or in our case

$$e^{\ln W} = \sum_{n=0}^{\infty} \frac{(\ln W)^n}{n!}$$

$$= 1 + (\ln W) + \frac{(\ln W)^2}{2!} + \ldots$$

Evaluating seven terms and expressing the results a little differently yields:

$$w = e^{\ln W} = 1 + \ln W \left(1 + \ln W \left(\frac{1}{2!} + \ln W \left(\frac{1}{3!} + \ldots \right.\right.\right.$$

$$\left.\left.\left. + \ln W \left(\frac{1}{6!} + \ln W \left(\frac{1}{7!}\right)\right) \cdots\right)\right.\right.$$

Hence:

$$X = W * 2^K$$

Calling Sequence:

```
SRJ        EXP, 17
```

Registers Used:

R4 = N (7 iterations)
R5 = Loop counter
R6 = lnW input Word 1
R7 = Input scale factor (2^K)
R10 = Word 1 output (double precision)
R11 = Word 2 output (double precision)
R12 = 1/n!

R14 = Temporary storage

R15 = Temporary storage

R17 = Exit register

Required Routines:

None

4.13 ADDR (Shown in Figure 4-13)

ADDR is a single precision routine written to evaluate the sum of two scaled input. Before the addition of any two scaled data inputs can be evaluated, they both must have the same scale factors. This is necessary since Word 2 of a scaled variable is actually a power of two by which Word 1 is scaled. Similar to polynomial addition, only like terms of the same power can be summed.

The input with the largest scaled factor is adjusted to the same scale factor of the other. The adjustments consist of performing the required number of right shifts to the smaller of the two inputs (i.e., the largest scale factor). Once both inputs have the same scale factor, the addition can be performed.

Algorithm:

Assume inputs: A_1, A_2 (WORD 1)

B_1, B_2 (WORD 2)

where

$$X_1 = A_1 * 2^{-B_1}$$

$$X_2 = A_2 * 2^{-B_2}$$

If $B_1 = B_2$, then

$$X_1 + X_2 = (A_1 + A_2) * 2^{-B_1}$$

If $B_1 < B_2$, then

$$X_1 = A_1 * 2^{-B_1}$$

$$X_2 = \left[A_2 * 2^{-(B_2 - B_1)}\right] * 2^{-B_1} = A_3 * 2^{-B_1}$$

where

$$A_3 = A_2 * 2^{-(B_2 - B_1)}$$

But since $B_2 - B_1 > 0$, then

$$A_3 = A_2 / 2^{B_2 - B_1}$$

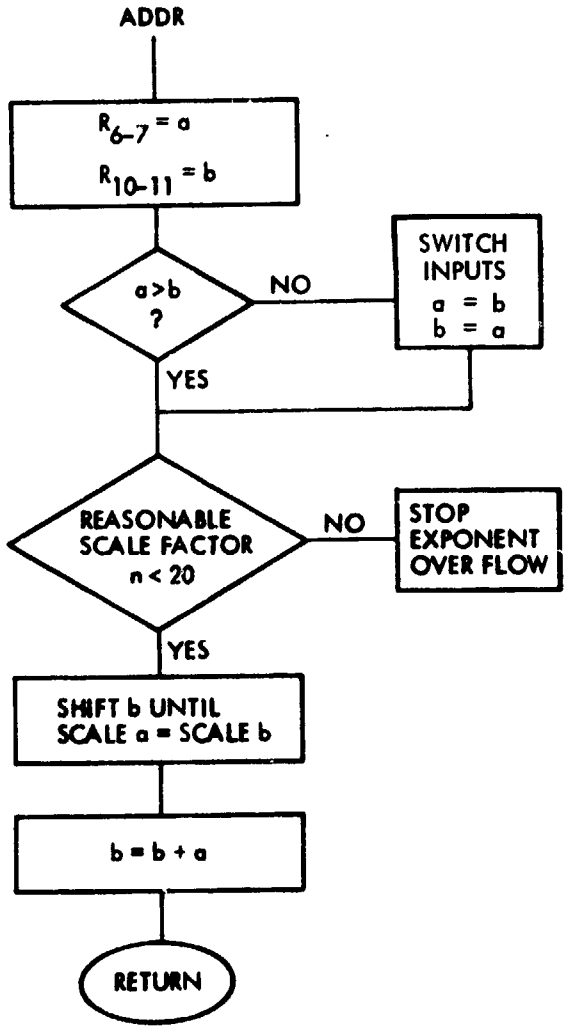


Figure 4-13. ADDR

Now one right shift of an octal number is equivalent to dividing by 2; then A3 is equal to A2 shifted to the right ($B_2 - B_1$) times and thus

$$X_1 + X_2 = (A_1 + A_3) * 2^{-B_1}$$

If $B_2 < B_1$, create a new variable such that

$A_1' = A_2$

$A_2' = A_1$

$B_1' = B_2$

$B_2' = B_1$

Now $B_1' < B_2'$ and the previous algorithm can be applied:

Example:

$A_1 = 23164 \qquad B_1 = 00004$

$A_2 = 16732 \qquad B_2 = 00002$

$X_1 = 23164 * 2^{-4}$

$X_2 = 16732 * 2^{-2}$

Performing two right shifts in X_1,

$$X_1 = 04635 * 2^{-2}$$

and

$$X_1 + X_2 = (04635 + 16732) * 2^{-2}$$
$$= (23567) * 2^{-2}$$

NOTE: For illustrative purposes the sign bit is omitted and assumed equal to zero (positive).

Calling Sequence:

```
SRJ     ADDR, 17
```

Registers Used:

R6 = First input Word 1
R7 = First input Word 2 (scale factor)
R10 = Second input Word 1/Result Word 1
R11 = Second input Word 2/Result Word 2
R12 = Temporary storage

Required Routines:

None

4.14 DIV (Shown in Figure 4-14)

DIV is a single precision division routine implemented for the purpose of dividing a two-word scaled data input by another two-word scaled data input.

Word 1 of both the divisor and dividend represent, as in all scaled data, a real number $0 < X < 1$; Word 2 represents the respective scale factor. Hence, Word 1 of the dividend can be divided by Word 1 of the divisor. Since the scale factors represent powers of 2, the quotient is evaluated by subtracting Word 2 of the divisor from Word 2 of the dividend.

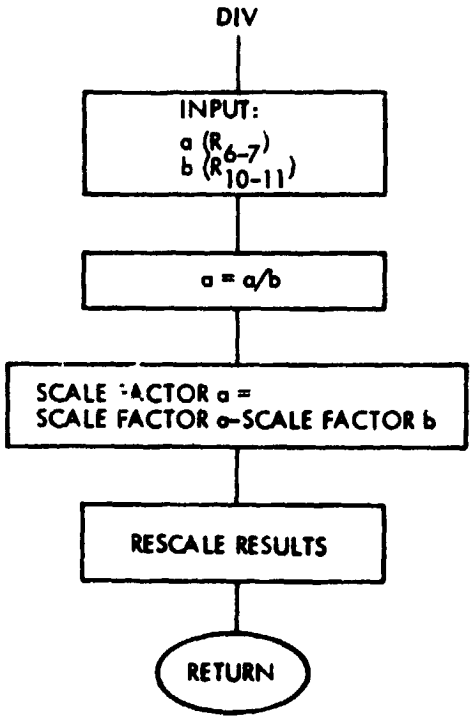


Figure 4-14. DIV

Algorithm:

Assume inputs: A_1, A_2 (WORD 1)

B_1, B_2 (WORD 2)

then,

$$X_1 = A_1 * 2^{-B_1}$$

$$X_2 = A_2 * 2^{-B_2}$$

and the quotient is

$$X_1/X_2 = (A1/A2) * 2^{-(B_1 - B_2)}$$

where the output WORD 1 = A_1/A_2 and the output WORD 2 = $(B_1 - B_2)$.

Calling Sequence:

```
SRJ        DIV, 17
```

Registers Used:

R6 = Dividend Word 1/Quotient Word 1

R7 = Dividend Word 2/Quotient Word 2

R10 = Divisor Word 1

R11 = Divisor Word 2

R12 = Temporary storage

R17 = Exit register

Required Routines:

SCALE

4.15 SCALE (Shown in Figure 4-15)

Since the CDC 469 mini-computer offers only five significant octal figures (a function of its 16-bit word), after many operations the results must be rescaled to retain maximum accuracy. This is accomplished by left shifting the result and dropping non-significant zero bits until a non-zero bit appears. Since real values are necessarily <1, only left shifts need be performed. The number of left shifts is recorded and becomes either the new scaling factor or a method of updating the old one.

This routine serves both an initial scaling function as well as a rescaling function. Initially, data are fed into the central processor as raw count data. External peripherals accumulate data as an integer result and feed this result into the processor as two words. This input must be converted to a scaled two-word data format. The first word represents a floating-point real number and the second represents a scale factor. This is accomplished by the process described above. Finally, the first five most significant bits are retained and stored into Word 1 and the number of left shifts is stored into Word 2.

The rescaling procedure is similar to the initial scaling sequence in its technique, but differs in two areas. First, the two-word input represents a previously scaled number, Word 1 being a real number and Word 2 a scale factor. Secondly, the number of left shifts is added to Word 2.

Both negative and positive numbers can be scaled or rescaled in this manner.

Example:

let $X = A * 2^{-B}$

where A = WORD 1 input = 03561

B = WORD 2 input = 0

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

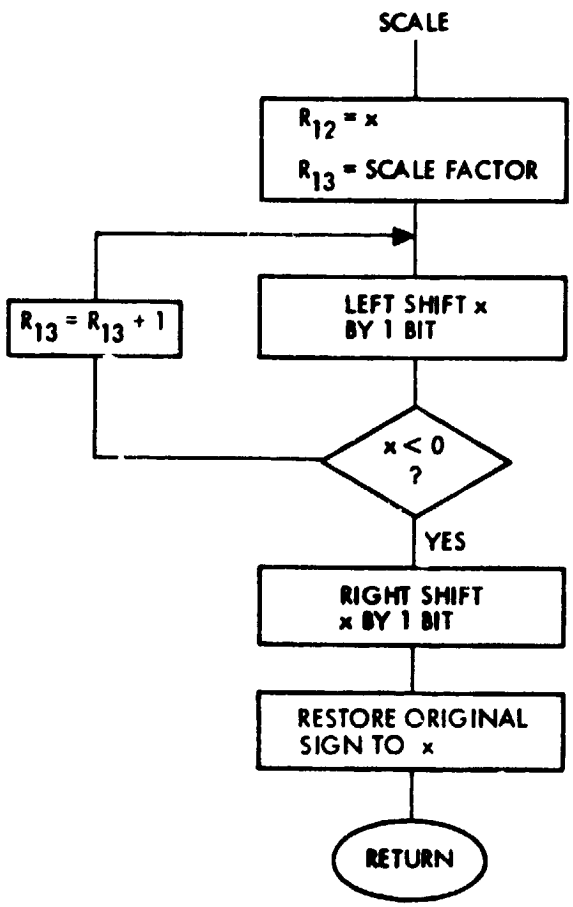


Figure 4-15. SCALE

The following table represents the values of A and B after each left shift.

Number of Left Shifts	Sign Bit	A	B
0	0	03561	0
1	0	07342	1
2	0	16704	2
3	0	25610	3
4	0	53420	4
5	1*	27040	5

*Denotes negative number.

Thus, the required scaling is four left shifts (B=4) for which A = 53420.

Calling Sequence:

- Initial Scaling:

```
SRJ     SCALE, 16
```

- Rescaling:

```
SRJ     SCALE + 2, 16
```

Registers Used:

R12 = Word 1 INPUT/SCALE FACTOR word output

R13 = Word 2 or scale FACTOR input/scale FACTOR output

R15 = Left shift counter

R16 = Exit register

4.16 SQRT (Shown in Figure 4-16)

SQRT is a single precision routine to generate the square root of a real input. The technique used is a recursive formula based on the rationale that if Y_i is an approximation to $Y = \sqrt{x}$, then

$$Y_{i+1} = \frac{1}{2}\left(Y_i + \frac{x}{Y_i}\right)$$

is a better approximation. Various simulation runs have shown that two iterations are sufficient for determining the square root of a 16-bit word (error $\geq 2^{-15}$).

Since Word 2 is the scaling factor for Word 1 and represents a power of 2, it must be adjusted by simple dividing by two, or equivalently performing a right shift by one bit. To insure an integer result, Word 2 should be input as an even integer. The adjustment, if Word 2 is odd, must also be made to Word 1 to preserve its initial value.

Algorithm: (Newton Iteration Technique)

let $X = A * 2^{-B}$

where A = WORD 1 input

B = WORD 2 input

then

$$X^{1/2} = A^{1/2} * 2^{-B/2}$$

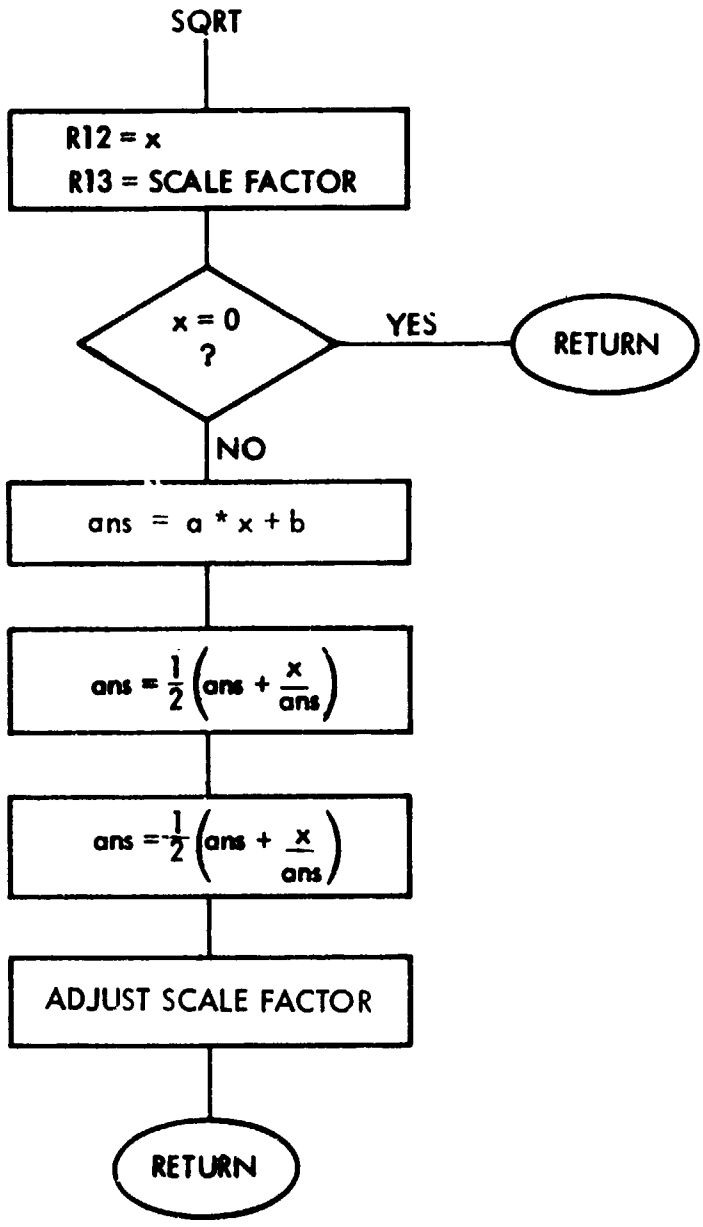


Figure 4-16. SQRT

B is first checked for an odd or even value to insure that B/2 is a whole number. In the event that B is odd, B is increased by unity and A is left-shifted one bit to preserve the initial value of X. If B is an even integer, no adjustment is required.

The first approximation for $A^{1/2}$ is the straight line Y = 0.6866 X + 0.3443; this is a good approximation over the interval $0.3 < X < 1.0$ and is particularly suited to the floating point arithmetic of the 469 computer; i.e., X is assuredly <1.0 and, if X is properly scaled, the most significant bit(s) are available and X >0.4. (Conversely, if the first bit is zero, then X <0.4 and rescaling is indicated.) Thus, with

$$Y_0 = 0.6866\ X + 0.3433$$

the first two iterations of the Newton technique yield

$$Y_1 = 1/2\ (Y_0 + A/Y_0)$$

$$A^{1/2} \approx Y_2 = 1/2\ (Y_1 + A/Y_1)$$

Calling Sequence:

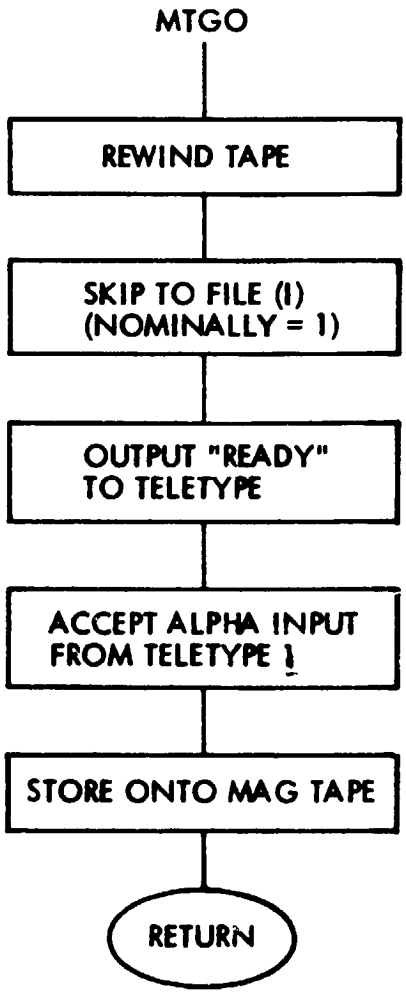
```
SRJ        SQRT, 17
```

Registers Used:

R0 = "P" Register
R6 = Temporary storage
R7 = Y(I)
R10 = X Input saved
R11 = Masking element
R12 = X Input/answer output
R13 = Scale factor input/scale factor output
R17 = Exit register

4.17 MTGO (Shown in Figure 4-17)

This routine is not part of the Zero-G programming task but is included both in the documentation and listing because of its data storage capability. MTGO performs the interface with a PERTEC 7000 magnetic tape unit. Logic has been incorporated into the program to automatically write on the magnetic tape the four channel accumulated count data along with the calculated mass. The format is the same as is printed on the teletype, and all data are octal. Computer channels four (4) and five (5) have been implemented for magnetic tape control, and the variable GOMT is the flag for writing or bypassing the write routines. The code generated now enters a zero into GOMT to perform the data storage as the program is running. Data are written at every 1-second interval.



1. INPUT IS TERMINATED BY TYPING "$"

Figure 4-17. MTGO

If the magnetic tape is no longer needed, the instruction at location 2023_8 (MTGO) should be changed as follows:

Core Location	Contents	Comment
2023_8	014000	Enables magnetic write
2023_8	014001	Bypasses magnetic tape routines

These routines have been incorporated into the program logic to facilitate data reduction for the mass versus "Z" length curvefits and can be eliminated at any time. If the mag routines are used and a print of the data is needed, only two operations are needed to read the tape. Register zero should be set equal to 6221_8 (MTRD) and register one set to the number of the first file to be read (octal). An end of file is written automatically each time the program is started from 2000_8.

The following is a list of the routines and locations that can be eliminated if so desired:

Routine Name		Core Locations (Octal)
MTGO		6000 - 6106
MTWRT		6107 - 6156
MTSTR		6157 - 6220
MTREAD		6221 - 6353
TEOF FTN READY WWAIT TFULL	Support Routines	6354 - 6477

4.18 PAUSE (Shown in Figure 4-18)

This routine will cause a pause in the program execution. This pause is susceptible to all three interrupt levels as well as the DX mode. Therefore, if the pause is to be independent of breaks, the interrupts must be locked out before entrance to this routine. The program will remain in this status until a bit is entered into the console's data register. The data register is initially cleared when the routine is called; therefore, any bit entered will terminate the pause and proceed with the program execution.

Calling Sequence:

```
SRJ     PAUSE, 17
```

Registers Used:

R10 = Console input

R17 = Exit register

4.19 STOPR (Shown in Figure 4-19)

During the execution of the program the data register on the console is checked after each mass calculation sequence for a non-zero input. If the input remains zero, the program continues generating new mass data. If, on the other hand, input is non-zero, the program discontinues mass calculations, writes an end-of-file on the mag-tape and pauses for further instructions from the console. At this point, two things may occur: either total program termination may result from a master clear; or the data bit

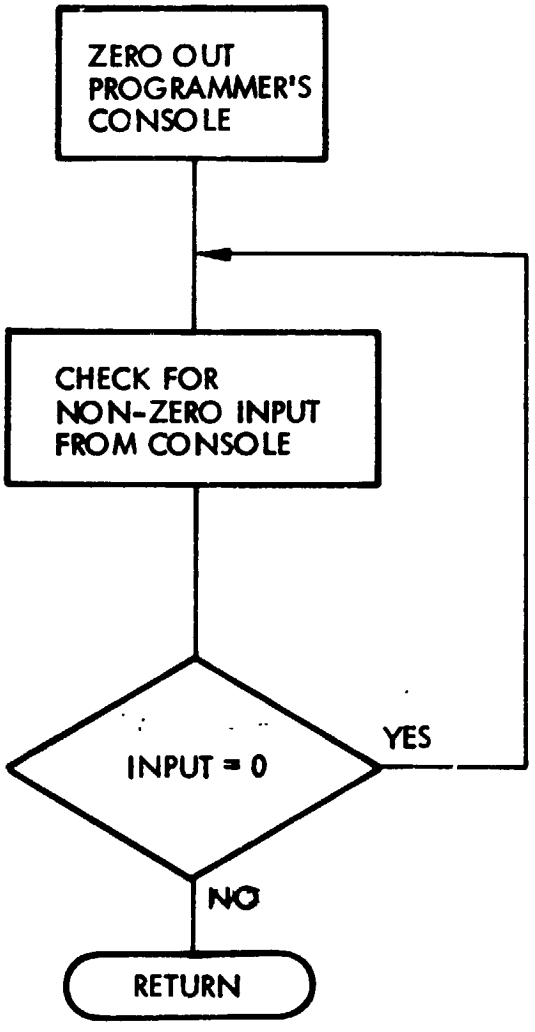


Figure 4-18. PAUSE

previously entered to halt mass calculation may be cleared, signifying the following sequence of events:

- Rewind tape to beginning of last file.
- Display on the teletype the contents of the last data file.
- Transfer control to the start of the original program.

At this time the program has been re-initialized and proceeds to print "ready" on the teletype and awaits description data input from the teletype. The program is now in the normal executing mode.

Calling Sequence:

```
UJ          STOPR
```

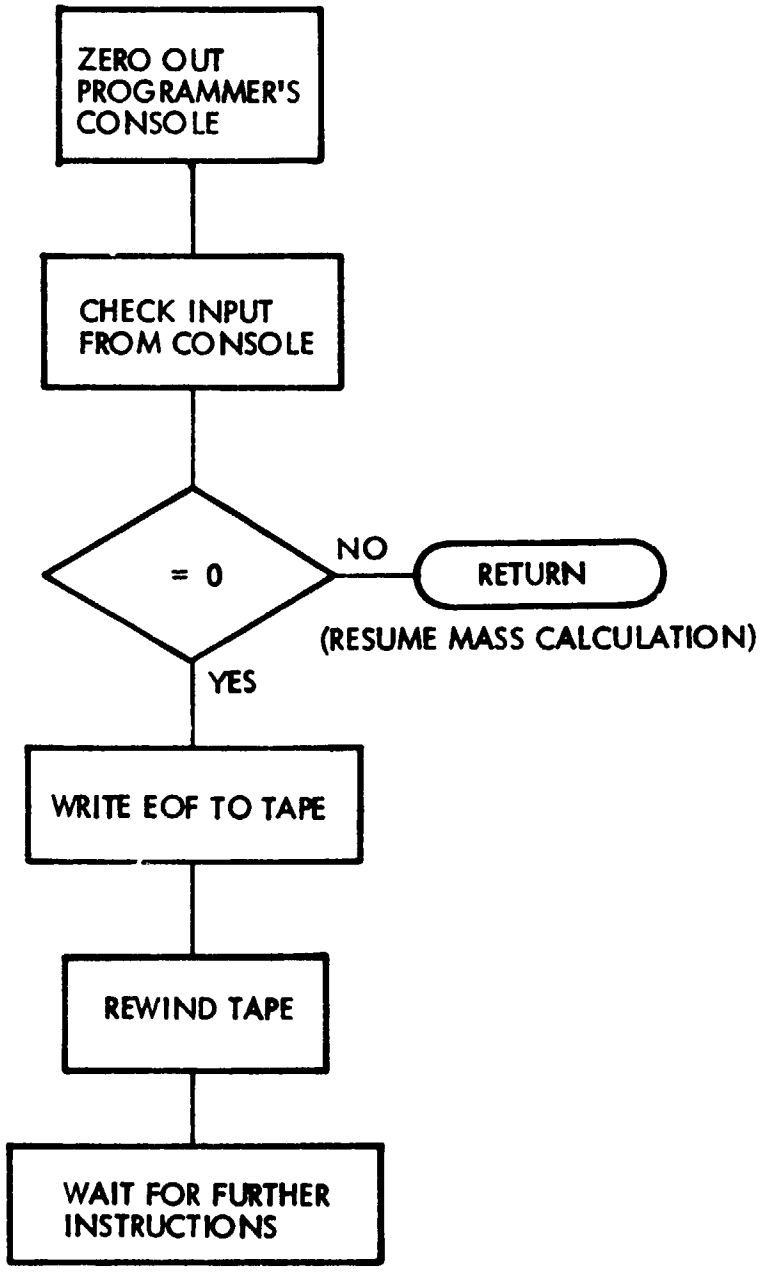


Figure 4-19. STOPR

Registers Used:

R1 = Argument register for MTREAD

R11 = Console inputs

4.20 MESG (Shown in Figure 4-20)

As the mnemonic implies the routine generates a message to the teletype. This message may be stored at any location in core and may

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

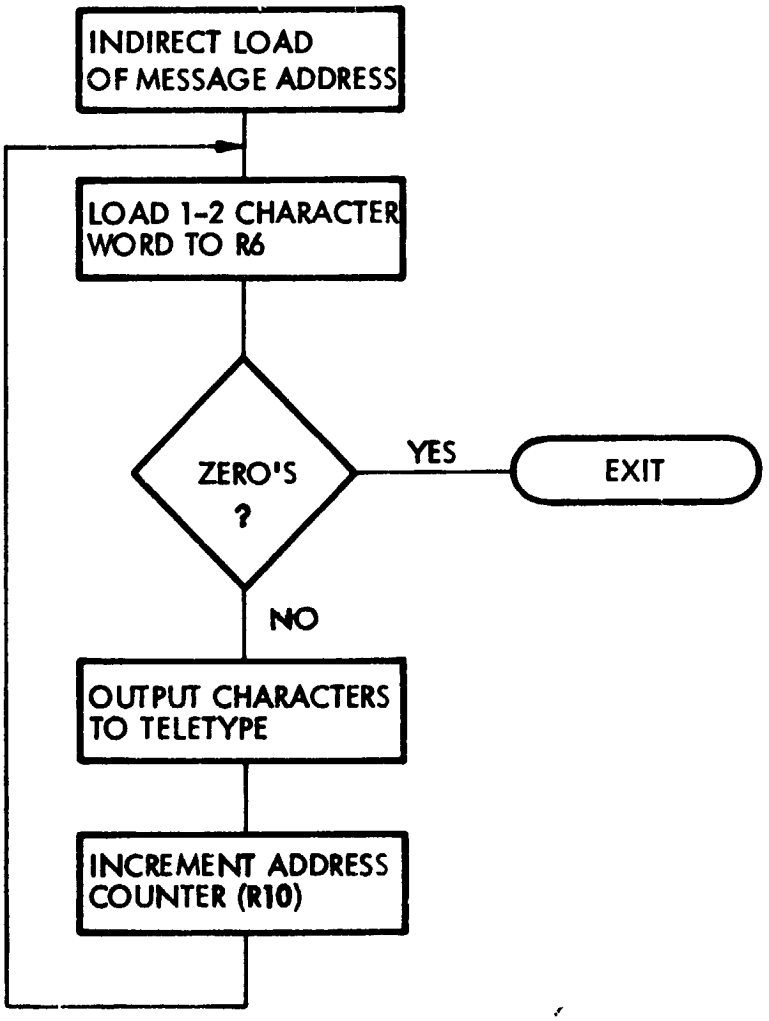


Figure 4-20. MESG

be entered either at compilation time or manually via the console. The form of input is an 8-bit ASC II code with two characters per word. A typical example is listed below:

```
17000   152325   TU
17001   151316   RN
17002   120317   - O
17003   143306   FF
17004   120314   - L     Turn off Load NDRO
17005   147701   OA
17006   142240   D -
17007   147304   ND
17010   151317   RO
```

The 8-bit ASC II code can be found in Table 5-1. The last word should be composed of all zeros to flag end of message.

Calling Sequence:

```
SRJ     MESG, 17
CON     MSG1        (location of first word of message – 17000₈)
```

Registers Used:

R6 = Character string

R7 = Teletype status

R10 = ADDR of message

R11 = Loop counter

R17 = Exit

4.21 ZCHK (Shown in Figure 4-21)

This routine provides a simple check on the system to verify counts from the detectors are being received. If all four of the input channels read zero counts, the display is blocked; the total mass is set to zero; and the status flag (word 200) is appropriately set to show system failure. A more sophisticated check is performed in FAIL. If any of the channels shows a non-zero input, the routine will exit and the program proceeds through its normal sequence of calls to generate the total mass.

Calling Sequence:

```
SRJ     ZCHK, 17
```

Registers Used:

R1 = Location of data inputs

R10 = Data

R11 = Zero (for comparison)

R12 = Counter

R17 = Exit

4.22 NES (Shown in Figure 4-22)

The gauging system, like any fine precision instrument, should be calibrated periodically. This is necessary for various reasons: (1) the source strength varies over a long period of time due to the natural radioactive decay; (2) the "alignment" of the source to the detector may change; (3) any object (insulation, supports, etc.) placed between the source and the detector will change the counts received. The recalibration is quite simple. With an empty tank, the gauging program is run for a few seconds (just long enough to gather a few data points for a meaningful sample).

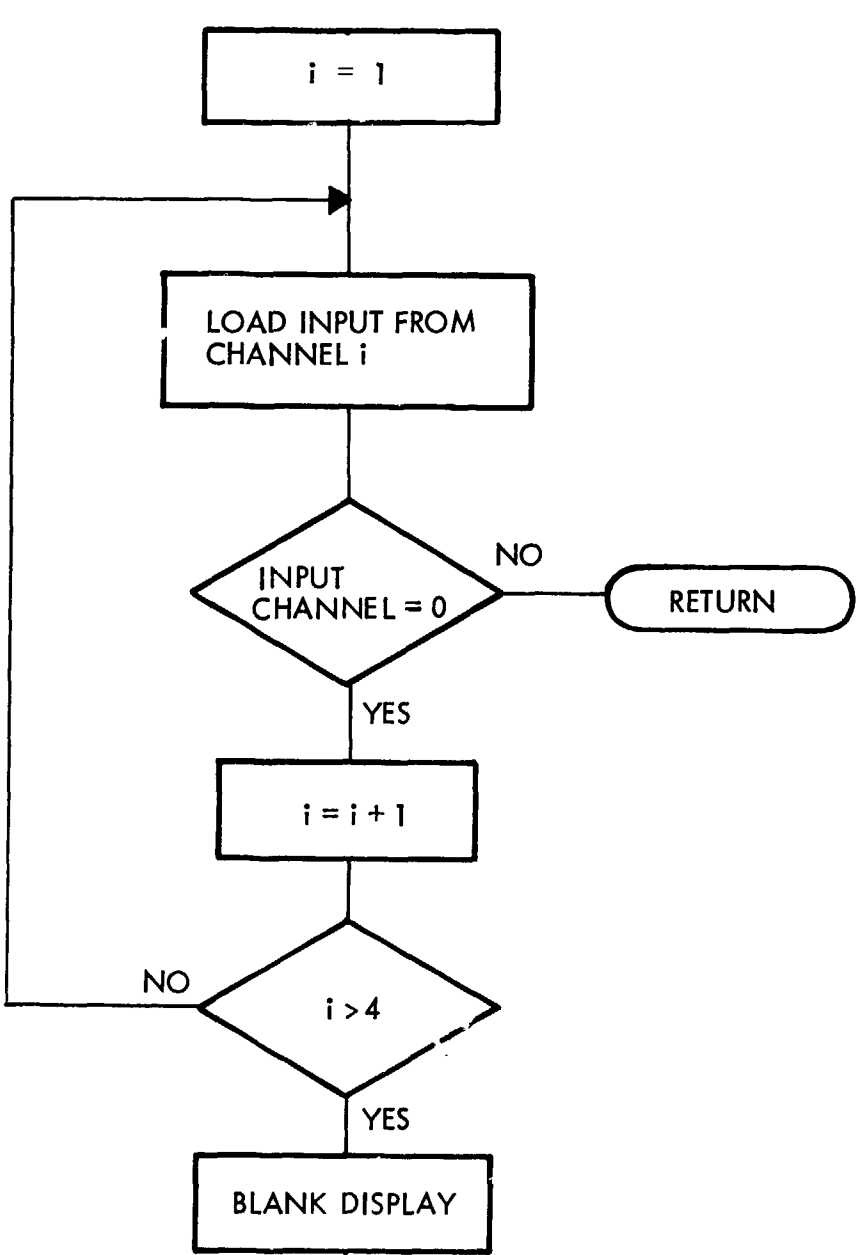


Figure 4-21. ZCHK

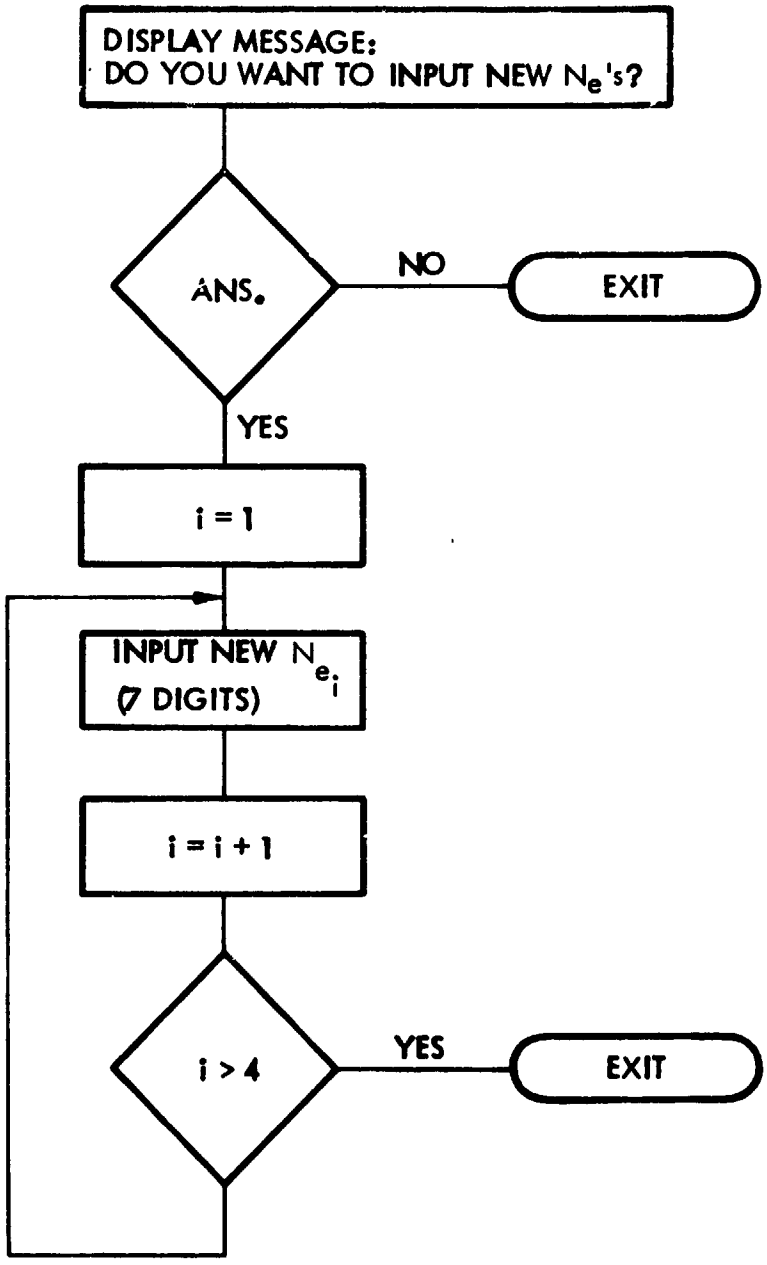


Figure 4-22. NES

The program is then stopped, the tape rewound and the data displayed on the teletype. The average of each column is the new N_e for the respective channel 1, 2, 3, or 4.

The program can then be restarted and the system will ask if new N_e's are to be entered. At this time the average counts just computed are input and the system is now recalibrated.

Calling Sequence:

```
SRJ        NES, 17
```

Registers Used:

R1 = Location of message
R4 = Counter
R5 = Counter
R6 = Input
R7 = Temporary
R10 = Teletype Status
R11 = Counter
R15 = 331B (Y – Yes)
R17 = Exit

4.23 IDENT (Shown in Figure 4-23)

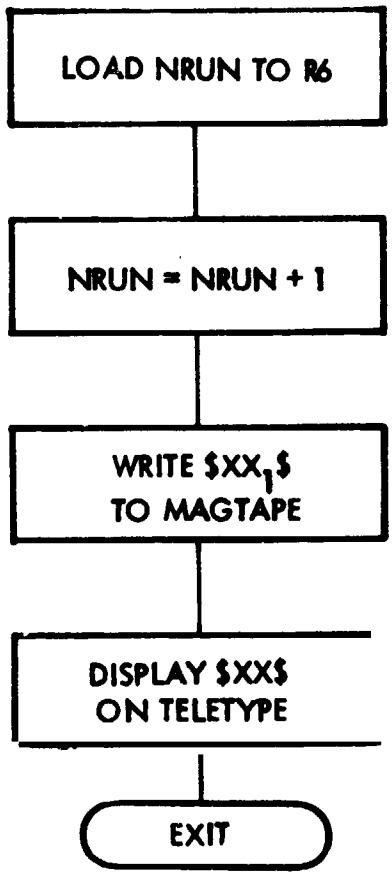
To distinguish one run from another, a software-generated numerical identifier is written directly onto the mag-tape as well as displayed on the terminal, at the start of each run. This was found necessary since all the runs are essentially the same format. This identifier automatically increments after each run and is encompassed by the '$' symbol for easy recognition on the printouts. The identifier is reset to zero during the initial program startup.

Calling Sequence:

```
SRJ        IDENT, 12
```

Registers Used:

R3 = Location of Counter (NRUN)
R4 = Counter
R5 = Temporary variable
R6 = NRUN
R7 = MASK
R10 = 260B – ASC (Zero)
R12 = Exit
R15 = 244B ($)



1 XX = NRUN

Figure 4-23. IDENT

4.24 PAUSR (Shown in Figure 4-24)

With any real time interactive system, such as the one for the gauge program, there exists the need to pause the system's computer. This is necessary since the computer is much faster than the response time necessary for the teletype to react to a command. The general use of PAUSR is to allow time for the teletype to perform a complete carriage return before the start of each message to be displayed. Therefore, this routine performs a number of CRS 20, 17 (circular right shift of R17, 20 bits) equivalent to 0.8 second. The CRS 20 instruction is essentially a NOP (NO-Operation), since to shift R17 20 bits is to leave it unchanged.

Calling Sequence:

```
SRJ     PAUSR, 17
```


REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

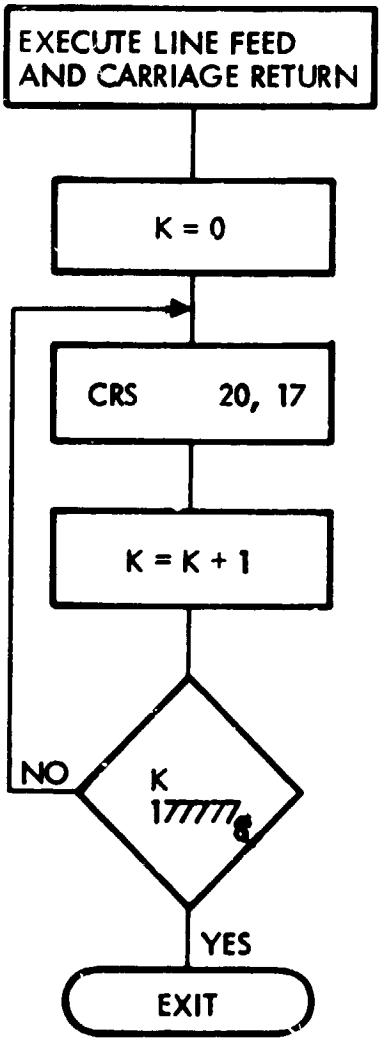


Figure 4-24. PAUSR

Registers Used:

R14 = Counter

R17 = Exit

Routines Called:

MESG

SECTION V

DATA REDUCTION

The Zero-G gauging system is comprised of three basic parts: (1) the source/detector pairs, (2) the computer, and (3) the interfacing hardware. The computer section can be further subdivided into three sections: data input, mass calculation, and data storage/output. The computer is controlled by an internal clock that issues interrupts at regular intervals of 1 second. Therefore, the entire cycle of data input, mass calculation, and data storage/output must be completed in less than 1 second. For this reason the only visible output during each cycle is the LED display of the final mass. Since output to a terminal is too time-consuming, all the necessary information for later data reduction is written to a mag-tape. Data input and the mass calculation have been discussed in the previous sections. It is the process of writing the information to the tape and the data reduction which is described in the following pages.

5.1 TAPE GENERATION

For purposes of data reduction and analysis, information collected and generated during each 1-second interval must be stored. Paper tape and/or hard copy generations are impossible because of the time limitations. Since disk options were not available, the only feasible means of data storage was writing to mag-tape. This tape generation offers a number of different methods of data reduction.

First, the tape can be stored for any length of time, remounted on the system, and printed on the local system teletype. Obviously, the tape can also be rewound at the conclusion of any flight and its contents displayed. This enables a quick visual inspection of the data for consistency as well as system status.

Secondly, the tape can be physically transferred to a large scale computer for major data reduction and evaluation. This option was found to be the most convenient and efficient way to do the proper analysis.

During each cycle of calculations, six values are written to mag-tape – the four channels of count information, a final mass, and a dummy variable (not used at this time). All six values are octal digits represented in standard 8-bit ASC code. (See Table 5-1.) It is the ASC representation that is actually written to the tape. Data are stored in this manner rather than octal or BCD for compatibility with the system teletype.

5.2 TAPE CONVERSION

The gauging system utilizes a Control Data 469 mini-computer for all logic and computations. This is a 16-bit machine with a core capacity of 8K. For data reduction, a large-scale Control Data 6500 is used. This is a 60-bit machine with a core capacity of 350K. The obvious difference

Table 5-1. ASC Code Conversion

Character	8-Bit Code (Octal)
A	301
B	302
C	303
D	304
E	305
F	306
G	307
H	310
I	311
J	312
K	313
L	314
M	315
N	316
O	317
P	320
Q	321
R	322
S	323
T	324
U	325
V	326
W	327
X	330
Y	331
Z	332
0	260
1	261
2	262
3	263

Table 5-1. ASC Code Conversion (Continued)

Character	8-Bit Code (Octal)
4	264
5	265
6	266
7	267
8	270
9	271
Line feed	215
Carriage return	212
Rub-out	377
Blank	240

is the word length, 16 versus 60 bits. A program, hereafter referred to as REDUCER, was written to convert the 16-bit word that was written to the mag-tape to the 60-bit word size in the larger computer.

Each tape generally consists of one day's flight, and each flight is composed of a number of maneuvers. It is during these maneuvers that the gauging system is activated, and data are generated. Each file on the tape corresponds to a particular maneuver. Each file is subdivided into a number of records. Each record represents one interval (1 second) of information. (See Figure 5-1.)

The first records of every file are dummy records containing a maneuver description in alphanumeric form. These records are used only for the system teletype and can be ignored by the large data reduction machine. Each subsequent record contains 9-60 bit words or equivalently 56_8 — 16 bit words, 3-1/3 - 16-bit words being packed into each 60-bit word. Each 16-bit word from the 469 is packed into one 18-bit segment of the larger 60-bit word as it is read from the tape. This is characteristic of data transfer between different word size machines since the transfer must be made for each octal number consisting of 3 bits. Therefore, 16 bits must be transferred to 18 bits, an even multiple of three. Zeros are placed in the remaining 2 bits. Terminating each record is an ASC code representing a teletype rub-out (377_8). These two conditions, the record format and the rub-out terminator, enable the appropriate unpacking.

As one ASC coded word (18-bits) is unpacked, it is decoded to one octal character. This character may either be a digit or a blank. If it is a digit, it is merged with any previous digit encountered after the last

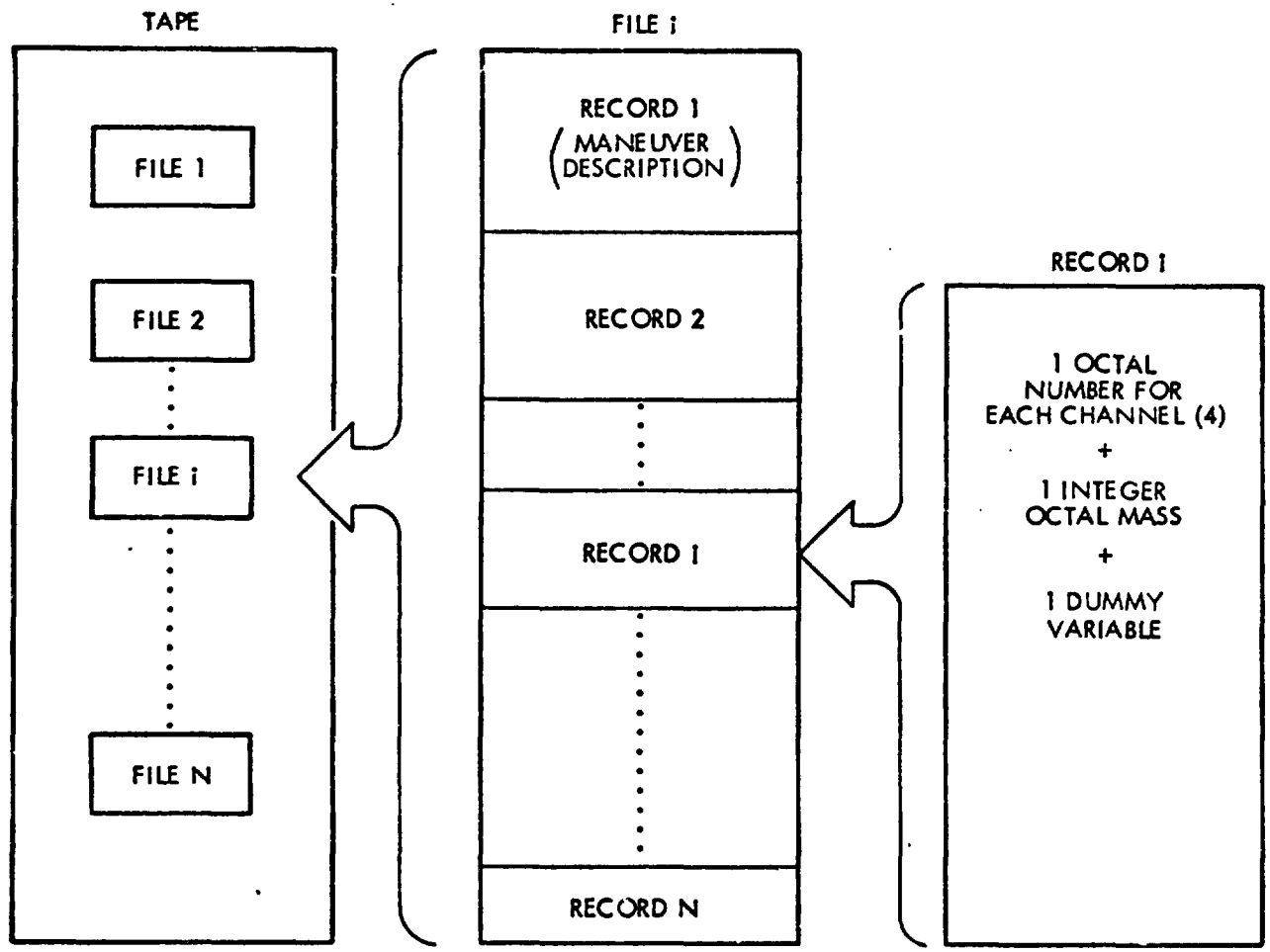


Figure 5-1. File Structure

blank. Therefore, all the digits between two blanks represent one piece of information. For example, the first sequence of six digits represent the counts received from channel 1 (as shown in Figure 5-2). At the completion of this unpacking and decoding sequence for this particular record, the six octal numbers are now as they were generated internally to the 469 mini-computer and are ready to be stored by the large machine. This entire process of unpacking and decoding has a net effect of converting the data written on the tape to the original format used by the 469, which is now compatible with the larger CDC machine.

This information is then written to a local file, denoted TAPE6, which is ultimately used as the basis for any data analysis to be performed. Figure 5-2 represents a dump of the beginning of a typical file (parabola), and Figure 5-3 represents a listing of the corresponding local file, i.e., TAPE6.

Figure 5-3 (TAPE6) represents more than six columns of numbers. The first six columns are the actual count data written to the tape. The next four columns represent the Z-length conversion corresponding to the count data for each channel per the following equation:

$$Z_i = \frac{1}{\mu} \ell n \left(\frac{N_i}{N_o} \right)$$

where

Z_i = the ray travel distance through the intervening mass (cm)

μ = mass absorption coefficient (cm^{-1})

N_i = number of counts received by channel $_i$

N_o = number of counts received in the absence of an intervening mass

Also note that the first six columns are octal while the last four are decimal.

5.3 REDUCTION

Once the data have been converted and stored on a file, virtually any analysis can be performed. A series of programs were written to employ different techniques of data evaluation. They are listed on the following pages, with a short description for each program.

Any references made to TAPE6 or TAPE7 are assumed to be the output files generated by REDUCER. TAPE6 is the total unedited data file, i.e., the actual conversion of the flight data. TAPE7 is a summary of TAPE6 and contains only the average values of the count data over each parabola and the number of 1-second gauging intervals (PTS) during each parabola (or 1-g gauging interval). See Figure 5-4.

```
(D469  )      FILE    2      REC       1            09/02/75. 11.20.48.

      0 *   CCCC C400 C215 0002 1200   C413 0002 1500 0000 0001

      2  WORDS IN RECORD.

(D469  )      FILE    2      REC       2            09/02/75. 11.20.48.

      3 *   CCC3 5200 C244 0002 6000   0263 0002 4400 C240 0002   6000 0260 0002 6300 0263   0002 7300 0271 0002 4000
      4 *   0313 CCC3 C1C0 0310 CCC4   4000 C2C1 0002 4000 0300   0003 1500 0320 0003 2400   0331 0002 1500 0212 0003
     10 *   77CC CCCC CCCC 0000 CCCC

     11  WORDS IN RECORD.

(D469  )      FILE    2      REC       3            09/02/75. 11.20.48.

      0 *   C300 5600 C260 0002 6000   0263 0002 6500 0267 0002   6500 0263 0002 4000 0260   0002 6000 0261 0002 6700
      4 *   0261 0002 6000 0261 0002   4000 0260 0002 6000 0262   0002 6400 0267 0002 6300   0260 0002 4000 0260 0002
     10 *   5000 0264 0002 6400 0262   0002 6200 0262 0002 4000   0260 0002 6000 0260 0002   6000 0260 0002 4000 0260
     14 *   0002 6000 0260 0002 6000   0260 0002 1500 0212 0003   7700 0000 0000 0000 0000

     17  WORDS IN RECORD.

(D469  )      FILE    2      REC       4            09/02/75. 11.20.48.

      0 *   0000 5600 0263 0002 6000   0263 0002 6100 0261 0002   6400 0260 0002 4000 0260   0002 6000 0261 0002 6500
      4 *   0267 0002 6500 0264 0002   4000 0260 0002 6000 0262   0002 6400 0262 0002 6700   0263 0002 4000 0260 0002
     10 *   0000 0264 0002 0000 0267   0002 6500 0260 0002 4000   0260 0002 6000 0260 0002   6000 0261 0002 4000 0260
     14 *   0002 6000 0260 0002 6000   0260 0002 1500 0212 0003   7700 0000 0000 0000 0000

     17  WORDS IN RECORD.

(D469  )      FILE    2      REC       5            09/02/75. 11.20.48.

      0 *   0000 5600 0260 0002 6000   0263 0002 6000 0267 0002   6200 0264 0002 4000 0260   0002 6000 0261 0002 6600
      4 *   0262 0002 6000 0263 0002   4000 0260 0002 6000 0262   0002 6400 0265 0002 6600   0261 0002 4000 0260 0002
     10 *   6000 0264 0002 6100 0266   0002 6400 0267 0002 4000   0260 0002 6000 0260 0002   6000 0260 0002 4000 0260
     14 *   0002 6000 0260 0002 6000   0260 0002 1500 0212 0003   7700 0000 0000 0000 0000

     17  WORDS IN RECORD.


(D469  )      FILE    2      REC       6            09/02/75. 11.20.48.

      0 *   0000 5600 0260 0002 6000   0263 0002 6100 0263 0002   6400 0263 0002 4000 0260   0002 6000 0261 0002 6500
      4 *   0266 0002 6200 0265 0002   4000 0260 0002 6000 0262   0002 6500 0267 0002 6100   0261 0002 4000 0260 0002
     10 *   6000 0263 0002 6700 0264   0002 6200 0267 0002 4000   0260 0002 6000 0260 0002   6000 0261 0002 4000 0260
     14 *   0002 6000 0260 0002 6000   0260 0002 1500 0212 0003   7700 0000 0000 0000 0000
```

Figure 5-2. Record Structure

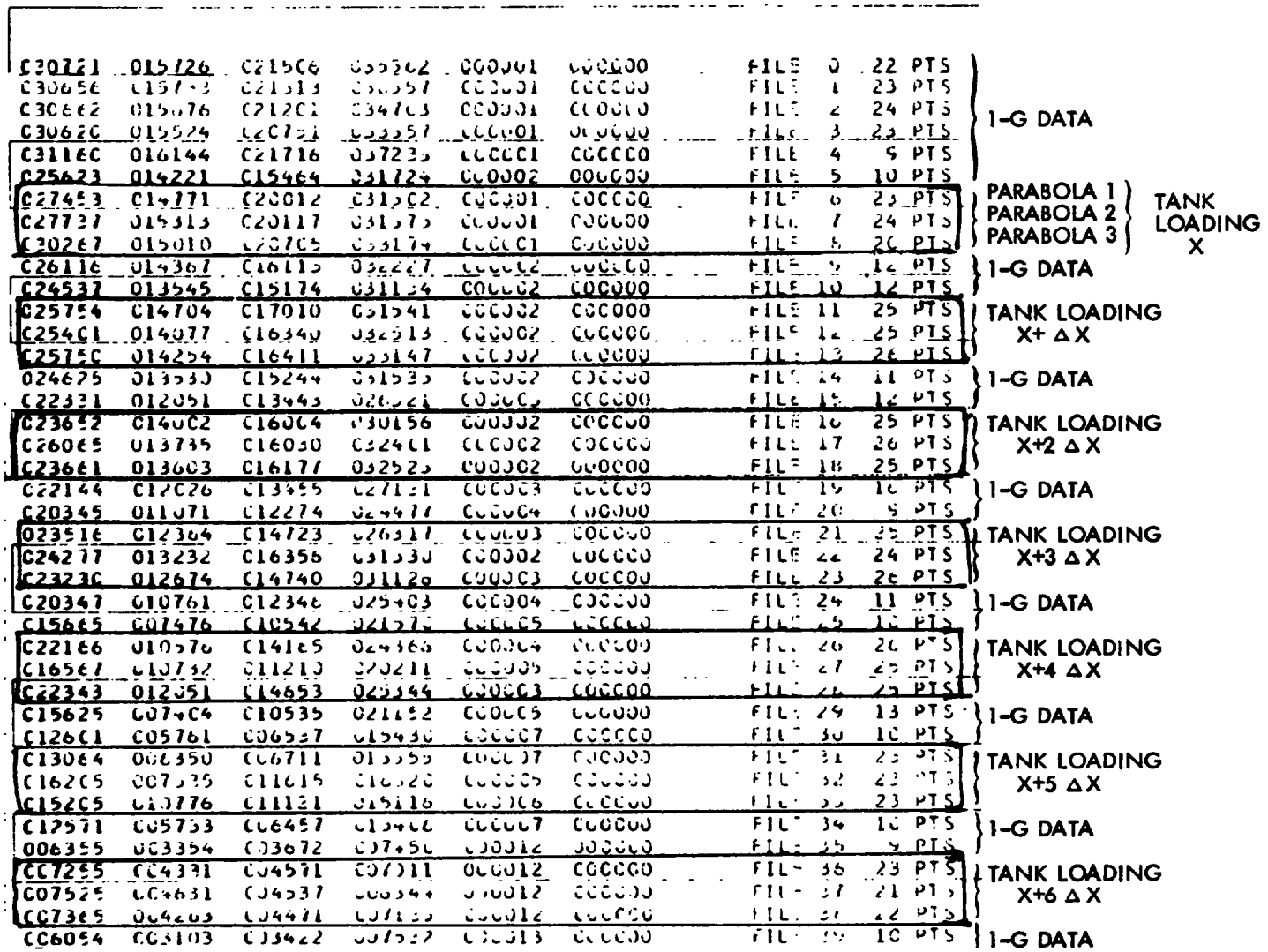
REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

FILE NO. 14

	Channel Number 1	2	3	4	Calculated Mass (lb)	Not Used	Z-Lengths (cm) for Channel 1	2	3	4
17	004695 1 G B1 2									
1	025777	014455	016032	033142	000002	000000	1.2	1.0	2.1	1.2
2	024570	012522	014533	026377	000003	000000	1.8	2.7	3.2	3.1
3	024757	013341	015226	030672	000003	000000	1.7	2.0	2.7	2.1
4	024766	013761	015553	032072	000002	000000	1.7	1.5	2.4	1.6
5	024513	013665	015343	035165	000002	000000	1.8	1.6	2.6	.5
6	025264	014076	015734	031726	000002	000000	1.5	1.4	2.3	1.7
7	024660	013347	015047	031451	000003	000000	1.8	2.0	2.9	1.8
8	025166	014322	015411	031364	000002	000000	1.6	1.2	2.5	1.9
9	024643	014034	015565	033017	000002	000000	1.8	1.4	2.4	1.3
10	024712	013474	015357	030374	000003	000000	1.7	1.8	2.6	2.3
11	022177	012171	013565	030100	000003	000000	3.1	3.1	3.9	2.4
FILE NO. 15										
20	004606 1 G B 3									
1	023467	012603	014354	026061	000003	000000	2.4	2.6	3.4	3.3
2	022372	011404	012710	024635	000004	000000	3.0	3.8	4.7	3.9
3	022267	012217	013540	027366	000003	000000	3.0	3.0	4.0	2.7
4	022743	012074	013561	025121	000004	000000	2.7	3.2	3.9	3.7
5	022324	012147	013525	027303	000003	000000	3.0	3.1	4.0	2.7
6	022347	012322	013550	026541	000003	000000	3.0	2.9	4.0	3.0
7	022245	012230	013500	026067	000004	000000	3.0	3.0	4.0	3.3
8	022301	012020	013456	030051	000003	000000	3.0	3.3	4.1	2.4
9	022213	012147	013603	026276	000003	000000	3.1	3.1	3.9	3.2
10	022535	012032	013510	030674	000003	000000	2.8	3.3	4.0	2.1

Figure 5-3. Sample Output Format of TAPE6

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR



C30721	015726	C21506	033562	000001	000000	FILE	0	22 PTS	1-G DATA
C30656	C15743	C21513	033557	000001	000000	FILE	1	23 PTS	
C30662	015676	C21201	034703	000001	000000	FILE	2	24 PTS	
C30620	015524	C20731	033557	000001	000000	FILE	3	25 PTS	
C3116C	016144	C21716	037235	000001	000000	FILE	4	5 PTS	
C25623	014221	C15464	031724	000002	000000	FILE	5	10 PTS	
C27453	C14771	C20012	031502	000001	000000	FILE	6	23 PTS	PARABOLA 1 / PARABOLA 2 / PARABOLA 3 — TANK LOADING X
C27737	015313	C20117	031575	000001	000000	FILE	7	24 PTS	
C30267	015010	C20705	033174	000001	000000	FILE	8	26 PTS	
C26116	014367	C16115	032227	000002	000000	FILE	9	14 PTS	1-G DATA
C24537	013545	C15174	031134	000002	000000	FILE	10	14 PTS	
C25754	C14704	C17010	031541	000002	000000	FILE	11	25 PTS	TANK LOADING X+ ΔX
C25401	014077	C16340	032513	000002	000000	FILE	12	25 PTS	
C25756	014254	C16411	032147	000002	000000	FILE	13	26 PTS	
024625	013330	C15244	031535	000002	000000	FILE	14	11 PTS	1-G DATA
C22331	012051	C13443	026021	000002	000000	FILE	15	14 PTS	
C23642	C14002	C16004	030156	000002	000000	FILE	16	25 PTS	TANK LOADING X+2 ΔX
C26065	013735	C16050	032401	000002	000000	FILE	17	26 PTS	
C23661	013603	C16177	032525	000002	000000	FILE	18	25 PTS	
C22144	C12C26	C13455	027151	000003	000000	FILE	19	10 PTS	1-G DATA
C20345	011071	C12274	024477	000004	000000	FILE	20	5 PTS	
023516	012364	C14723	026517	000003	000000	FILE	21	25 PTS	TANK LOADING X+3 ΔX
C24277	013232	C16356	031530	000002	000000	FILE	22	24 PTS	
C23236	012674	C14740	031120	000003	000000	FILE	23	26 PTS	
C20347	010761	C12346	025403	000004	000000	FILE	24	11 PTS	1-G DATA
C15665	007476	C10542	021575	000005	000000	FILE	25	10 PTS	
C22166	010570	C14165	024365	000004	000000	FILE	26	26 PTS	TANK LOADING X+4 ΔX
C16567	010742	C11210	020411	000005	000000	FILE	27	25 PTS	
C22343	012051	C14653	026544	000003	000000	FILE	28	25 PTS	
C15625	007404	C10535	021452	000005	000000	FILE	29	13 PTS	1-G DATA
C12601	005761	006527	015430	000007	000000	FILE	30	10 PTS	
C13064	006350	C06711	015355	000007	000000	FILE	31	25 PTS	TANK LOADING X+5 ΔX
C16205	007535	C11615	022520	000005	000000	FILE	32	25 PTS	
C15205	010776	C11131	025116	000006	000000	FILE	33	23 PTS	
C12571	005753	C06457	015406	000007	000000	FILE	34	10 PTS	1-G DATA
006355	003354	C03672	007450	000012	000000	FILE	35	5 PTS	
CC7255	CC4391	C04571	C07011	000012	000000	FILE	36	23 PTS	TANK LOADING X+6 ΔX
C07525	004631	C04537	000344	000012	000000	FILE	37	21 PTS	
CC7365	004203	C04471	007155	000012	000000	FILE	38	22 PTS	
CC6054	003103	C03422	007527	000013	000000	FILE	39	10 PTS	1-G DATA

Figure 5-4. TAPE7 Sample Format

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

5.3.1 REDUCER

This program converts the tape format of the actual flight data tape (generated by the CDC 469) to the tape format required by the CDC 6000 series computer. Two local files – denoted TAPE6 and TAPE7 – contain the resulting output of the conversion (as shown in Figures 5-3 and 5-4).

5.3.2 BUPP

This program operates on the data of TAPE6 (generated by REDUCER) and calculates the average values and standard error of estimates for the Z-length for a running sequence of gauging data throughout each parabola. For example, two integers, N_1 and N_2, are requested at the start of execution. The first integer, N_1, defines at which 1-second gauging interval data are to be considered, whereas the second integer, N_2, defines at which 1-second gauging interval beyond which data are not considered. In short, N_1 and N_2 define a sequence of consecutive 1-second gauging intervals upon which the following operations are performed:

$$\bar{Z}_j = \frac{\sum_{i=N_1}^{N_2} Z_{ij}}{N_2 - N_1 + 1}$$

$$\text{Standard Error of Estimate} = \left[\frac{\sum_{i=N_1}^{N_2} \left(Z_{ij} - \bar{Z}_j\right)^2}{(N_2 - N_1 + 1) - 1} \right]^{\frac{1}{2}}$$

After the indicated operations, N_1 and N_2 are then each advanced by unity and the operations repeated to form the "running" sequence until N_2 equals the last 1-second gauging interval of that parabola.

5.3.3 BUPP4

This program operates on the data of TAPE6 and calculates the average value and the standard error of estimate for the Z-lengths over one fixed consecutive sequence of 1-second gauging intervals. BUPP4 was used primarily to delete the early and late portions of the parabola where the propellant was in a (near) one-g configuration and provided the "summary" data used as input to ZEROFIT.

5.3.4 ZEROFIT

The program which relates the calculated Z-lengths to the actual mass of propellant within the tank is denoted ZEROFIT. ZEROFIT is basically a curve-fitting program which utilizes an orthogonal polynomial fit by a least squares approximation; i. e., the error $(M_{actual} - M_{curve})^2$

is minimized by successive approximations to

$$M = A_o + A_1 Z + A_2 Z^2 + \cdots$$

The coefficients, A_n, so determined from ZEROFIT are ther stored in the CDC 469 so that propellant mass can then be calculated in real time in the flight environment.

SECTION VI

SYSTEM COMPONENTS

The following major components are required for the development and implementation of the software for the Zero-G operating system. (Figure 5-1 presents a functional relationship between these components.)

- Control Data Corporation 469 (I) Micro-computer
- Pertec Magnetic Tape Unit 7X20
  a) 7 track
  b) 7-inch reel
  c) 25 ips
  d) 556 Bpi
- Control Data Corporation Programmer Console

  ID No. 460-02 Series A
  Part No. 69012100-02 Series 9R
- TI 733 KSR W/RS232 Interface (or ASR 33 TTY)
- Teletype Interface
- Cathode Ray Tube – BALL (MIRATEL)

6.1 DIGITAL PROCESSOR

The Control Data 460 Digital Computer is a fourth-generation, state-of-the-art metal oxide semiconductor (MOS)/large scale integrated (LSI) digital computer designed to operate reliably in areas of applications where small size, light weight, low power, and existence in a severe environment are imperative. Physical and electrical characteristics of the 469 computer are presented in Table 6-1.

6.2 MEMORY

The memory used in the 469 processor is constructed with plated wire elements and is typically referred to as PWM (plated wire memory). This type of memory allows random access, is word organized, and does not require rewriting after a read cycle. Because of this nondestructive readout characteristic, the operating mode of the system is much faster. The basic word length is 16 bits with a read cycle time of 1.6 microseconds, a write cycle time of 2.4 microseconds, and an access time of 900 nanoseconds.

Table 6-1. CDC 469 Computer Description

Word Length	16 bits
Memory Size	8, 192 words
Interface Levels	0 to 5 volts
Weight	2. 75 pounds
Volume	60 cubic inches
Power	15 watts
Cooling	Conduction and Radiation

Central Processor

- General Purpose, Binary, Parallel, Single Address
- Logic: High Level P-MOS/LSI
- Hardware Registers: 16
- Arithmetic: Fractional, Fixed Point, Two's Complement
- Clock: 1 MHz
- Instructions: 42 Total

Input/Output

- Input Channels: ONE, 16 bit, parallel
- Output Channels: ONE, 16 bit, parallel
- Interrupt Levels: 3 Plus Direct Execute
- Serial I/O: 1 IN: 1 OUT

6.3 CENTRAL PROCESSOR

The operational registers for the 469 central processor reside in a set of MOS/LSI devices called the register file. This register file contains sixteen 16-bit registers composed of four MOS/LSI devices, each containing eight words of 8 bits each. The file registers are addressable as the first 16 words of memory for upward references. However, instructions cannot be executed from the register file.

The central processor of the 469 computer has a basic word length (arithmetic register) of 16 bits. For double precision, the word length is 32 bits (two arithmetic registers). All arithmetic instructions operate in a two's complement, fixed point, fractional mode. The instruction repertoire is made up of 42 instructions, of which 41 are 16 bits long and one instruction, SRJ (subroutine jump), is 32 bits long. This instruction repertoire consists of single and double precision add and subtract, single precision fractional multiply and divide, logical operations, load, store, single and double precision shifts, subroutine jump, unconditional jumps, interrupt control, input/output (I/O), and functional decisions (conditional jumps and skips).

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

Register assignment is a function of the current interrupt level. At a minimum level 0, register "zero" is committed as the program address (P) register, and register "one" is selected for the index register. Register zero for the P register is mandatory, although any register can be defined as the normal mode index register by using the load index select (LIS) instruction. There exist three interrupt levels as well as a direct execute (DX) level.

Figure 6-1 presents the functional relationship of Control Data Corporation 469 computer and the peripherals requisite to software development.

6.4 PROGRAMMER'S CONSOLE

The programmer's console is connected directly to the central processor and displays instantaneously the contents of the register files. The user has the option of controlling the execution of the object code either under a step-by-step execution mode or a normal execution mode. If a certain instruction need be changed, the new instruction and the related data word can be entered onto the console and entered directly into the object code from the console.

6.5 TI 733 KSR

For larger code modifications the 733 KSR can be used. This teletype, similar to the terminals available for TRW/TSS timesharing, generally has the function of controlling the entire system. The teletype is connected on a controller interface apparatus that provides the necessary interface between the more universal teletype unit and the specific central processor.

6.6 MAG-TAPE

The PERTEC magnetic tape drive serves two purposes. First, via the CDC 6600, a compiled version of the program can be loaded directly into central memory for later execution by the console. Secondly, once the program is operational, the count data generated by each channel and the total system mass can be stored on the magnetic tape for each selected counting interval. These stored data can then later be batch-processed to aid in the data reduction and/or curve fit purposes for calibration.

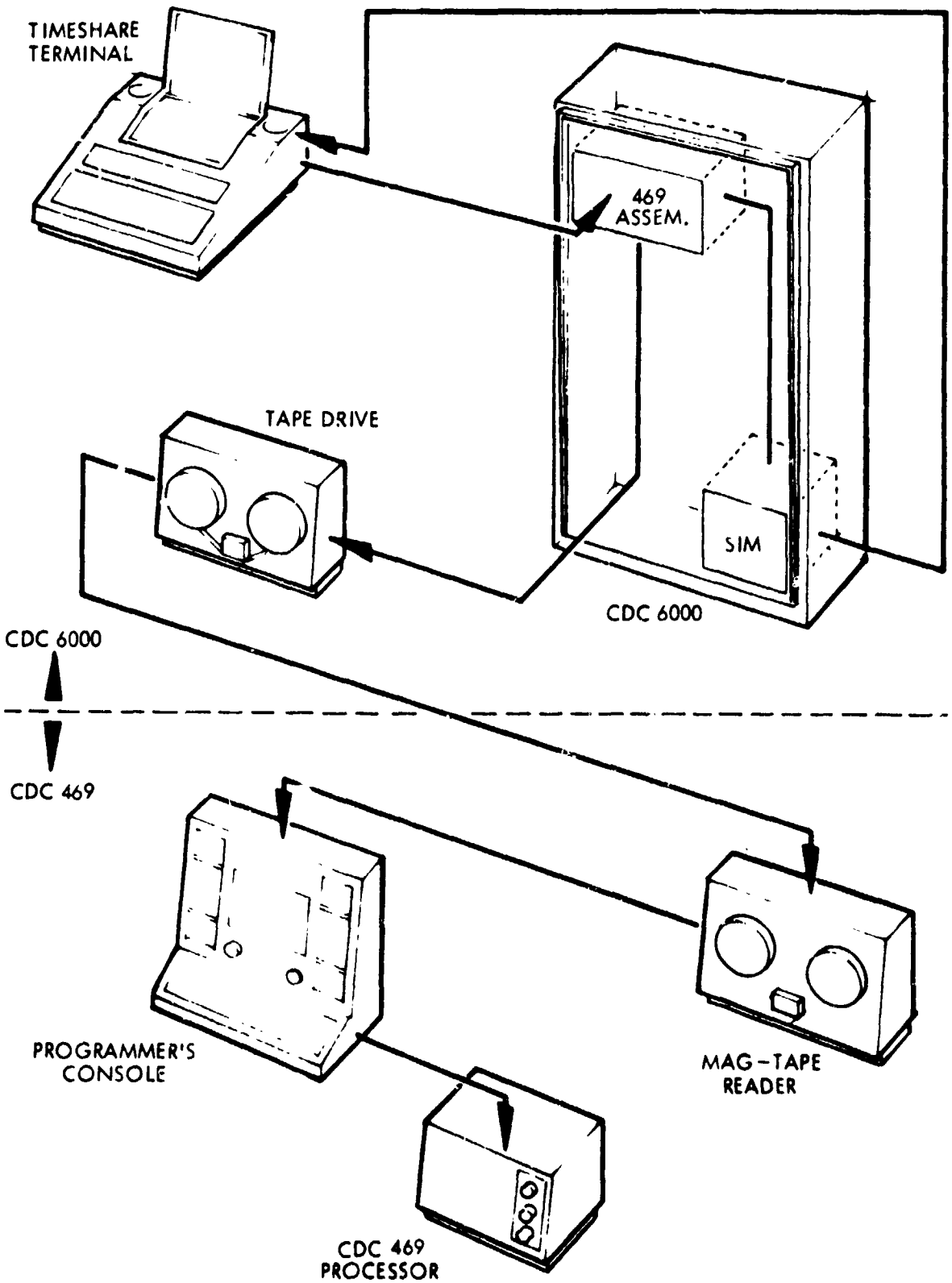


Figure 6-1. Functional Relationship of Zero-Gauge Peripherals

REFERENCES

1. "The Development of a Zero-G Gauging System," Final Report No. 16740-6003-RU-00, 20 January 1974.

2. 469 Computer Programming (CDC Reference Manual), CDC.

3. 469 Computer Simulator Reference Manual (Version 1.0), CDC.

4. 469 Computer Assembler Reference Manual (Version 1.0), CDC.